
ECP Service Provider Edition API Documentation

Release 3.0.4

Enomaly

January 20, 2010

CONTENTS

1	Authentication	3
1.1	Introduction	3
1.2	Examples	3
2	billing – Billing	7
2.1	Resources	7
2.2	Representations	7
3	clone – Clone virtual machines.	9
3.1	Resources	9
3.2	Representations	9
4	dtemplate – Disk Template	11
4.1	Resources	11
4.2	Representations	12
4.3	Examples	13
5	group – Group	17
5.1	Resources	17
5.2	Representations	18
6	htemplate – Hardware Template	21
6.1	Resources	21
6.2	Representations	22
6.3	Examples	24
7	log – Logs and Transactions	27
7.1	Resources	27
7.2	Representations	28
7.3	Examples	29
8	network – Network	31
8.1	Resources	31
8.2	Representations	32
8.3	Examples	33
9	ptemplate – Package Template	37
9.1	Resources	37
9.2	Representations	38
9.3	Examples	38

10	quota – Resource Quota	41
10.1	Resources	41
10.2	Representations	43
11	tag – Tagging	45
11.1	Resources	45
11.2	Representations	46
12	usage – Virtual Machine Resource Usage	49
12.1	Resources	49
12.2	Representations	50
13	user – User	51
13.1	Resources	51
13.2	Representations	52
14	util – Utilities	55
14.1	Resources	55
14.2	Representations	56
15	vm – Virtual Machine	59
15.1	Resources	59
15.2	Representations	61
16	Indices and tables	63
	Module Index	65
	Index	67

Welcome to the ECP Service Provider Edition API documentation. Here you will find resources on how to use the ECP Service Provider Edition RESTful API.

AUTHENTICATION

1.1 Introduction

The ECP API uses basic HTTP authorization in order to establish a connection with clients. To establish a connection, the client needs to send an initial HTTP request containing the username and password. If authentication succeeded, the client will receive a non-error HTTP response code and response message.

The initial HTTP login request can be either a *GET* request or a *POST* request.

1.2 Examples

1.2.1 PHP

Here is an example of how to establish a connection by sending a *GET* request.

```
1 <?php
2
3 //Basic connection parameters.
4 $base_url = '127.0.0.1';
5 $username = 'username';
6 $password = 'password';
7
8 //The cookie file used to store the session cookie.
9 $cookies_file = 'cookies.txt';
10
11 //Initialize the client object.
12 $client = curl_init();
13
14 //Do some basic client configuration. We need to be able to read cookies
15 //from a file. Since we are authenticating to initialize a connection,
16 //we don't necessarily care about the response body.
17 curl_setopt($client, CURLOPT_RETURNTRANSFER, 1);
18 curl_setopt($client, CURLOPT_COOKIEFILE, $cookies_file);
19 curl_setopt($client, CURLOPT_COOKIEJAR, $cookies_file);
20 curl_setopt($client, CURLOPT_HEADER, 1);
21 curl_setopt($client, CURLOPT_NOBODY, 1);
22
23 //Initialize the authentication GET URL including authentication parameters.
24 curl_setopt($client, CURLOPT_URL, $base_url."?user_name=$username&password=$password&login=Login");
25
```

```
26 //Send the request and initialize the response code.
27 $http_response = curl_exec($client);
28 $http_response_code = curl_getinfo($client, CURLINFO_HTTP_CODE);
29
30 //Close the connection.
31 curl_close($client);
32
33 //Test the result of the authentication attempt.
34 if ($http_response_code==401 || $http_response_code==0){
35     echo "Authentication failed.";
36 }
37 else{
38     echo "Authentication sucessful.";
39 }
40
41 ?>
```

Here is an example of how to establish a connection by sending a *POST* request.

```
1 <?php
2
3 //Basic connection parameters.
4 $base_url = '127.0.0.1';
5 $username = 'username';
6 $password = 'password';
7
8 //The cookie file used to store the session cookie.
9 $cookies_file = 'cookies.txt';
10
11 //Initialize the client object.
12 $client = curl_init();
13
14 //Do some basic client configuration. We need to be able to read cookies
15 //from a file. Since we are authenticating to initialize a connection,
16 //we don't necessarily care about the response body.
17 curl_setopt($client, CURLOPT_URL, $base_url);
18 curl_setopt($client, CURLOPT_RETURNTRANSFER, 1);
19 curl_setopt($client, CURLOPT_COOKIEFILE, $cookies_file);
20 curl_setopt($client, CURLOPT_COOKIEJAR, $cookies_file);
21 curl_setopt($client, CURLOPT_HEADER, 1);
22 curl_setopt($client, CURLOPT_NOBODY, 1);
23 curl_setopt($client, CURLOPT_POST, true);
24
25 //Initialize the authentication POST parameters including authentication parmeters.
26 curl_setopt($client, CURLOPT_POSTFIELDS, "user_name=$username&password=$password&login=Login");
27
28 //Send the request and initialize the response code.
29 $http_response = curl_exec($client);
30 $http_response_code = curl_getinfo($client, CURLINFO_HTTP_CODE);
31
32 //Close the connection.
33 curl_close($client);
34
35 //Test the result of the authentication attempt.
36 if ($http_response_code==401 || $http_response_code==0){
37     echo "Authentication failed.";
38 }
39 else{
```

```
40     echo "Authentication sucessful.";
41 }
42
43 ?>
```


BILLING – BILLING

Billing functionality.

2.1 Resources

2.1.1 Billing

class Billing()

This resource represents a query into the available billing data for users and the virtual machine resources used by those users.

Resource URI:

`/modules/billingreports/`

GET (*start_date*, *end_data*)

Retrieve the virtual machine resource usage data in CSV format by submitting an *HTTP GET* request.

Note: This resource requires *read* permission on both the virtual machine and the *elastic_hosting* extension module when using the *HTTP GET* method.

Parameters

- *start_date* – The starting date of the resource usage data (YYYY:MM:DD:HH:mm:ss).
- *end_data* – The ending data of the resource usage data (YYYY:MM:DD:HH:mm:ss).

Return type `billing.RepBilling`

2.2 Representations

2.2.1 RepBilling

class RepBilling()

This is a representation of the virtual machine resource usage for all users of the system.

Note: The first row in this representation is descriptive of the following rows. In this case, there are three hardware profiles. The number corresponding to each hardware profile, for each user row, indicates the number of hours that user has consumed the virtual machine resources required by that profile. The last column is disk space measured in MB.

Representation:

```
user, user_uuid, Small, Medium, Large, totalMBytes  
customer1, 7eabba12-12fe-11dd-8fd0-0019d2b28af0, 0, 0, 0, 0  
customer2, 09492bbc-3fd3-11de-82cf-001a929face2, 0, 3, 0, 1024
```

CLONE – CLONE VIRTUAL MACHINES.

Virtual machine cloning functionality.

3.1 Resources

3.1.1 Clone

class Clone ()

This resource represents a method in which to clone a virtual machine.

Resource URI:

`/rest/hosting/clone/<UUID>`

POST ()

Clone the specified virtual machine by submitting an *HTTP PUT* request. The *UUID* in the resource URI should be the UUID of the target virtual machine.

Note: This resource requires *create* permission on the *elastic_hosting* extension module.

Return type `vm.RepClone`

See Also:

`vm`

3.2 Representations

3.2.1 RepClone

class RepClone ()

This is a representation of a virtual machine being cloned.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general message generated by the resource.

machine_id

This attribute represents the UUID of the virtual machine being cloned.

Representation:

```
{"errno": response.errno,  
 "message": response.message,  
 "machine_id": vm.uuid}
```

DTEMPLATE – DISK TEMPLATE

Disk template functionality.

4.1 Resources

4.1.1 DtemplateUUID

class DtemplateUUID ()

This resource represents a specific disk template.

Resource URI:

`/rest/hosting/dtemplate/<UUID>/`

GET ()

Retrieve the specified disk template by submitting an *HTTP GET* request. The *UUID* in the resource uri should be the uuid of the target disk template.

Note: This resource requires *read* permission on both the disk template and the *elastic_hosting* extension module when using the *HTTP GET* method.

Return type `dtemplate.RepDtemplateUUID`

4.1.2 DtemplateList

class DtemplateList ()

This resource represents a list of disk templates.

Resource URI:

`/rest/hosting/dtemplate/list/`

GET ()

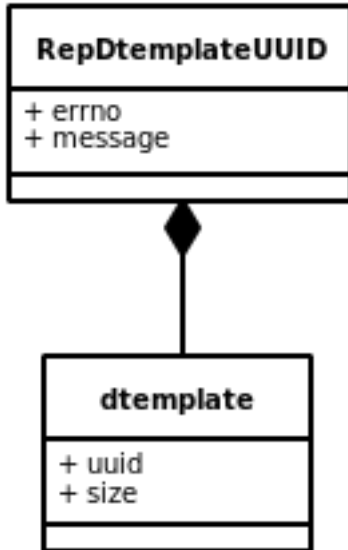
Retrieve a list of disk templates by submitting an *HTTP GET* request.

Note: This resource requires *read* permission on the *elastic_hosting* extension module.

Return type `dtemplate.RepDtemplateList`

4.2 Representations

4.2.1 RepDtemplateUUID



class RepDtemplateUUID ()

This is a representation of a disk template resource.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general message generated by the resource.

uuid

This attribute represents the *ID* of the disk template resource.

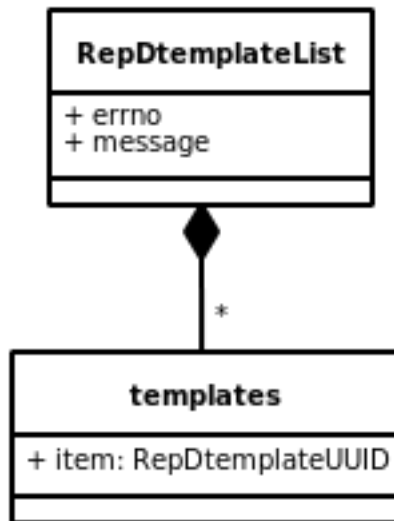
size

This attribute represents the *size* of the disk template resource. This attribute is measured in *MB*.

Representation:

```
{"errno": response.errno,
 "message": response.message,
 "dtemplate": {"uuid": dtemplate.uuid,
               "size": dtemplate.size}}
```

4.2.2 RepDtemplateList



class RepDtemplateList ()

This is a representation of a list of disk templates. This is a list of `dtemplate.RepDtemplateUUID` representations.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general error message generated by the resource.

templates

This attribute represents the list of template representations.

Representation:

```

{"errno": response.errno,
 "message": response.message,
 "templates": [dtemplate1, dtemplate2]}
  
```

4.3 Examples

4.3.1 PHP

Here is an example of how to list all available disk templates.

```

1 <?php
2
3 //Make an API call to retrieve all disk templates.
4 $disks = ecp_api_get($client, $base_url.'/rest/hosting/dtemplate/list/', 'templates');
5
6 //Make sure there was a response.
7 if($disks){
8
9     //Display all returned disk data.
  
```

```
10     echo '<h1>Disk templates:</h1>';
11     echo '<ol>';
12     foreach ($disks as $disk){
13         echo "<li>".($disk['size']/1024)." GB</li>";
14     }
15     echo '</ol>';
16 }
17 else{
18     echo 'No disk templates available!';
19 }
20
21 //Basic function to make an API request.
22 function api_request($client, $url, $field){
23
24     //Make the request and initialize the HTTP response.
25     curl_setopt($client, CURLOPT_URL, $url);
26     $http_response = curl_exec($client);
27
28     //Make sure the API responded.
29     if($http_response){
30
31         //Initialize the response code.
32         $http_response_code = curl_getinfo($client, CURLINFO_HTTP_CODE);
33
34         //Make sure the HTTP request succeeded.
35         if($httpcode == 200){
36
37             //Initialize the JSON object.
38             $json_response = json_decode($http_response, true);
39
40             //Make sure the JSON object was initialized.
41             if($json_response){
42
43                 //Make sure the internal response code was successful.
44                 if($json_response['errno'] === 0){
45
46                     //Retrieve the specified field data.
47                     $field_data = $json_response[$field];
48
49                     //Make sure the data exists and return it.
50                     if($data){
51                         return $field_data;
52                     }
53                     else{
54                         echo 'Error: No data available.';
55                     }
56                 }
57                 else{
58                     echo 'Error: '.$response_arr['message'];
59                 }
60             }
61             else{
62                 echo 'Error: Response could not be parsed.';
63             }
64         }
65         else{
66             echo 'Error: Request failed.';
67         }
68     }
69 }
```

```
68     }
69     else{
70         echo 'Error: No response.';
71         return false;
72     }
73 }
74
75 ?>
```


GROUP – GROUP

Group functionality.

5.1 Resources

5.1.1 Group

class Group ()

This resource represents a method in which to create a new group.

Resource URI:

`/rest/hosting/group/`

PUT (*groupname*, *displayname*)

Create a new group by submitting an *HTTP PUT* request.

Note: This resource requires *create* permission on the *elastic_hosting* extension module.

Parameters

- *groupname* – New group name.
- *displayname* – New group display name.

Return type `group.RepGroup`

5.1.2 GroupUUID

class GroupUUID ()

This resource represents a specific group.

Resource URI:

`/rest/hosting/group/<UUID>/`

GET ()

Retrieve the specified group by submitting an *HTTP GET* request. The *UUID* in the resource URI should be the UUID of the target group.

Note: This resource requires *read* permission on both the group and the *elastic_hosting* extension module when using the *HTTP GET* method.

Return type `group.RepGroupUUID`

POST (*groupname*, *displayname*)

Update the specified group by submitting an *HTTP POST* request. The *UUID* in the resource URI should be the *UUID* of the target group.

Note: This resource requires *update* permission on both the group and the *elastic_hosting* extension module when using the *HTTP POST* method.

Parameters

- *groupname* – The groupname to update the group with.
- *displayname* – The display name to update the group with.

Return type `group.RepGroupUUID`

5.1.3 GroupList

class `GroupList` ()

This resource represents a list of groups.

Resource URI:

`/rest/hosting/group/list/`

GET ()

Retrieve a list of groups by submitting an *HTTP GET* request.

Note: This resource requires *read* permission on the *elastic_hosting* extension module and on each group that is found.

Return type `group.RepGroupList`

5.2 Representations

5.2.1 RepGroup

class `RepGroup` ()

This is a representation of a group resource.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general message generated by the resource.

group

This attribute represents the newly created group object and is a `group.RepGroupUUID` representation.

Representation:

```
{"errno": response.errno,  
 "message": response.message,  
 "group": group_obj}
```

5.2.2 RepGroupUUID

class RepGroupUUID ()

This is a representation of a group resource.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general message generated by the resource.

uuid

This attribute represents the UUID of the group.

group_name

This attribute represents the groupname of the group.

display_name

This attribute represents the display name of the group.

Representation:

```
{ "errno": response.errno,
  "message": response.message,
  "user": { "uuid": group.uuid,
            "group_name": group.group_name,
            "display_name": group.display_name } }
```

5.2.3 RepGroupList

class RepGroupList ()

This is a representation of a list of groups. This is a list of user .RepGroupUUID representations.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general message generated by the resource.

groups

This attribute represents the list of group representations.

Representation

```
{ "errno": response.errno,
  "message": response.message,
  "groups": [group1, group2] }
```

HTEMPLATE – HARDWARE TEMPLATE

Hardware template functionality.

6.1 Resources

6.1.1 Htemplate

class `Htemplate` ()

This resource represents a method in which to create a new hardware template.

Resource URI:

```
/rest/hosting/htemplate/
```

PUT ()

Create a new hardware template by submitting an *HTTP PUT* request.

Note: This resource requires *create* permission on the *elastic_hosting* extension module.

Parameters

- *name* – New hardware template name.
- *arch* – New hardware template architecture.
- *hypervisor* – New hardware template hypervisor.
- *memory* – New hardware template memory.
- *cpus* – New hardware template cpus.

Return type `htemplate.RepHtemplate`

6.1.2 HtemplateUUID

class `HtemplateUUID` ()

This resource represents a specific hardware template.

Resource URI:

```
/rest/hosting/htemplate/<UUID>/
```

GET ()

Retrieve the specified hardware template by submitting an *HTTP GET* request. The *UUID* in the resource URI should be the UUID of the target hardware template.

Note: This resource requires *read* permission on both the hardware template and the *elastic_hosting* extension module when using the *HTTP GET* method.

Return type `quota.RepHtemplateUUID`

POST ()

Update the specified hardware template by submitting an *HTTP POST* request. The *UUID* in the resource URI should be the UUID of the target hardware template.

Note: This resource requires *update* permission on both the hardware template and the *elastic_hosting* extension module when using the *HTTP POST* method.

Parameters

- *name* – The name of the hardware template.
- *arch* – The cpu architecture of the hardware template.
- *hypervisor* – The hypervisor of the hardware template.
- *memory* – The memory of the hardware template..
- *cpus* – The number of cpus for the hardware template.

Return type `htemplate.RepHtemplateUUID`

DELETE ()

Delete the specified hardware template by submitting an *HTTP DELETE* request. The *UUID* in the resource URI should be the UUID of the target hardware template.

6.1.3 HtemplateList

class HtemplateList ()

This resource represents a list of hardware templates.

Resource URI:

`/rest/hosting/htemplate/list/`

GET ()

Retrieve a list of hardware templates by submitting an *HTTP GET* request.

Note: This resource requires *read* permission on the *elastic_hosting* extension module.

Return type `htemplate.RepHtemplateList`

6.2 Representations

6.2.1 RepHtemplate

class RepHtemplate ()

This is a representation of a new hardware template that has been created.

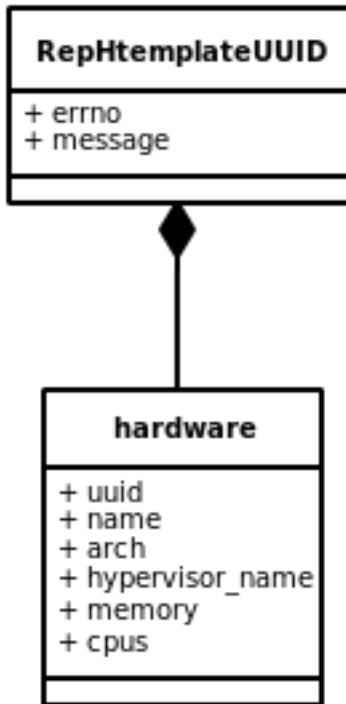
Representation:

```

{"errno": response.errno,
 "message": response.message,
 "hardware": hardware}

```

6.2.2 RepHtemplateUUID



class RepHtemplateUUID ()

This is a representation of a specific hardware template.

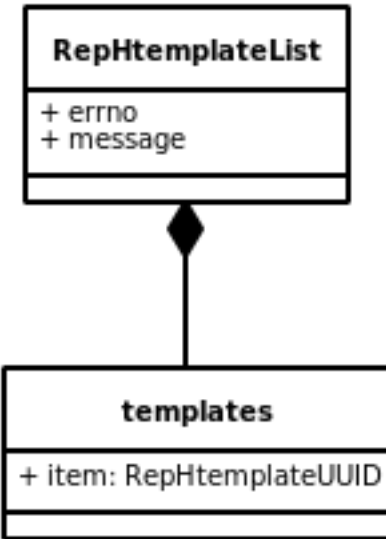
Representation:

```

{"errno": response.errno,
 "message": response.message,
 "hardware": {"uuid": hardware.uuid,
              "name": hardware.name,
              "arch": hardware.arch,
              "hypervisor_name": hardware.hvm,
              "memory": hardware.memory,
              "cpus": hardware.cpus}}

```

6.2.3 RepHtemplateList



class RepHtemplateList ()

This is a representation of a list of hardware templates.

Representation:

```

{"errno": response.errno,
 "message": response.message,
 "templates": [htemplate1, htemplate2]}
  
```

6.3 Examples

6.3.1 PHP

Here is an example of how to list all available hardware templates.

```

1 <?php
2
3 //Make an API call to retrieve all hardware templates.
4 $hardware = ecp_api_get($client, $base_url.'/rest/hosting/htemplate/list/', 'templates');
5
6 //Make sure there was a response.
7 if($hardware) {
8
9     //Display all returned hardware data.
10    echo '<h1>Hardware templates:</h1>';
11    echo '<ol>';
12    foreach ($hardware as $_hardware) {
13        echo "<li>$_hardware[name]: $_hardware[description]</li>";
14    }
15    echo '</ol>';
16 }
17 else {
18    echo 'No hardware templates available!';
  
```

```
19 }
20
21 //Basic function to make an API request.
22 function api_request($client, $url, $field){
23
24     //Make the request and initialize the HTTP response.
25     curl_setopt($client, CURLOPT_URL, $url);
26     $http_response = curl_exec($client);
27
28     //Make sure the API responded.
29     if($http_response){
30
31         //Initialize the response code.
32         $http_response_code = curl_getinfo($client, CURLINFO_HTTP_CODE);
33
34         //Make sure the HTTP request succeeded.
35         if($httpcode == 200){
36
37             //Initialize the JSON object.
38             $json_response = json_decode($http_response, true);
39
40             //Make sure the JSON object was initialized.
41             if($json_response){
42
43                 //Make sure the internal response code was successful.
44                 if($json_response['errno'] === 0){
45
46                     //Retrieve the specified field data.
47                     $field_data = $json_response[$field];
48
49                     //Make sure the data exists and return it.
50                     if($data){
51                         return $field_data;
52                     }
53                     else{
54                         echo 'Error: No data available.';
55                     }
56                 }
57                 else{
58                     echo 'Error: '.$response_arr['message'];
59                 }
60             }
61             else{
62                 echo 'Error: Response could not be parsed.';
63             }
64         }
65         else{
66             echo 'Error: Request failed.';
67         }
68     }
69     else{
70         echo 'Error: No response.';
71         return false;
72     }
73 }
74
75 ?>
```


LOG – LOGS AND TRANSACTIONS

Logging and transactional functionality.

7.1 Resources

7.1.1 LogUUID

class LogUUID ()

This resource represents a specific action log.

Resource URI:

```
/rest/hosting/log/<UUID>/
```

GET ()

Retrieve the specified action log by submitting an *HTTP GET* request. The *UUID* in the resource URI should be the uuid of the target action log.

..note

This resource requires **read** permission on the **elastic_hosting** extension module.

Return type `log.RepLogUUID`

7.1.2 LogList

class LogList ()

This resource represents a list of action logs and running transactions.

Resource URI:

```
/rest/hosting/log/list/
```

GET ()

Retrieve a list of action logs and running transactions by submitting an *HTTP GET* request.

..note

This resource requires **read** permission on the **elastic_hosting** extension module.

Return type `log.RepLogList`

7.2 Representations

7.2.1 RepLogUUID

class RepLogUUID ()

This is a representation of an action log.

user

This attribute represents the *user* id of the log.

timestamp

This attribute represents the *timestamp* attribute of the log. This is the time at which the log was created.

error

This attribute represents the *error* attribute of the log. The value of this attribute will be true if the log is an error message.

message

This attribute represents the *message* attribute of the log. This message is the main content of the log.

Representation:

```
{"user": log.user,  
 "timestamp": log.timestamp,  
 "error": log.error,  
 "message": log.message}
```

7.2.2 RepLogList

class RepLogList ()

This is a representation of a list of action logs and running transactions.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general message generated by the resource.

logs

This attribute represents the list of action log representations.

transactions

This attribute represents the list of transaction representations.

Representation:

```
{"errno": response.errno,  
 "message": response.message,  
 "logs": [log1, log2],  
 "transactions": [transaction1, transaction2]}
```

7.3 Examples

7.3.1 PHP

Here is an example of how to monitor running transactions.

```
1 <html>
2 <head>
3     <title>Running Transactions</title>
4     <meta http-equiv="refresh" content="5">
5 </head>
6 <body>
7     <h1>Running Transactions:</h1>
8
9     <?php
10    //Get the list of running transactions.
11    $transactions = ecp_api_get($client, $base_url.'/rest/hosting/log/list/', 'transactions');
12
13    //Make sure there are transactions to display and display them.
14    if($transactions) {
15        echo '<ul>';
16        foreach ($transactions as $transaction) {
17            echo '<li>'. $transaction['completion'].'% '. $transaction['message'].'</li>';
18        }
19        echo '</ul>';
20    }
21    else{
22        echo 'No pending transactions.';
23    }
24    ?>
25
26 </body>
27 </html>
```


NETWORK – NETWORK

Network functionality.

8.1 Resources

8.1.1 Network

class Network ()

This resource represents a method in which to create a new virtual network.

Resource URI:

`/rest/hosting/network/`

PUT (*name, vlan_id*)

Create a new virtual network by submitting an *HTTP PUT* request.

Note: This resource requires *create* permission on the *elastic_hosting* extension module.

Parameters

- *name* – New virtual network name.
- *vlan_id* – The vlan ID this network is associated with.

Return type `network.RepNetwork`

8.1.2 NetworkUUID

class NetworkUUID ()

This resource represents a specific virtual network.

Resource URI:

`/rest/hosting/network/<UUID>/`

GET ()

Retrieve the specified virtual network by submitting an *HTTP GET* request. The *UUID* in the resource URI should be the UUID of the target virtual network.

Note: This resource requires *read* permission on the *elastic_hosting* extension module.

Return type `network.RepNetworkUUID`

POST (*name*, *vlan_id*)

Update the specified virtual network by submitting an *HTTP POST* request. The *UUID* in the resource URI should be the UUID of the target virtual network.

Note: This resource requires *read* permission on the *elastic_hosting* extension module.

Parameters

- *name* – The virtual network name.
- *vlan_id* – The vlan ID this network is associated with.

Return type `network.RepNetworkUUID`

DELETE ()

Delete the specified virtual network by submitting an *HTTP DELETE* request. The *UUID* in the resource URI should be the UUID of the target virtual network.

8.1.3 NetworkList

class NetworkList ()

This resource represents a list of virtual networks.

Resource URI:

```
/rest/hosting/network/list/
```

GET ()

Retrieve a list of virtual networks.

Note: This resource requires *read* permission on the *elastic_hosting* extension module.

Return type `RepNetworkList`

8.2 Representations

8.2.1 RepNetwork

class RepNetwork ()

This is a representation of a newly created virtual network.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general message generated by the resource.

network

This attribute represents the newly created network.

Representation:

```
{ "errno": response.errno,  
  "message": response.message,  
  "network": network_obj }
```

8.2.2 RepNetworkUUID

class RepNetworkUUID ()

This is a representation of a specific virtual network.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general message generated by the resource.

uuid

This attribute represents the UUID of the virtual network.

name

This attribute represents the name of the virtual network.

vlan_id

This attribute represents the vlan ID of the virtual network.

Representation:

```
{ "errno": response.errno,
  "message": response.message,
  "network": { "uuid": network.uuid,
               "name": network.name,
               "vlan_id": network.vlan_id } }
```

8.2.3 RepNetworkList

class RepNetworkList ()

This is a representation of a list of virtual networks.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general message generated by the resource.

networks

This attribute represents the list of virtual network representations.

Representation:

```
{ "errno": response.errno,
  "message": response.message,
  "networks": [network_obj1, network_obj2] }
```

8.3 Examples

8.3.1 PHP

Here is an example of how to list all available networks.

```
1 <?php
2
3 //Make an API call to retrieve all disk templates.
4 $disks = ecp_api_get($client, $base_url.'/rest/hosting/network/list/', 'networks');
5
6 //Make sure there was a response.
7 if($networks){
8
9     //Display all returned network data.
10    echo '<h1>Networks:</h1>';
11    echo '<ol>';
12    foreach ($networks as $network){
13        echo "<li>$item[name]</li>";
14    }
15    echo '</ol>';
16 }
17 else{
18     echo 'No networks available!';
19 }
20
21 //Basic function to make an API request.
22 function api_request($client, $url, $field){
23
24     //Make the request and initialize the HTTP response.
25     curl_setopt($client, CURLOPT_URL, $url);
26     $http_response = curl_exec($client);
27
28     //Make sure the API responded.
29     if($http_response){
30
31         //Initialize the response code.
32         $http_response_code = curl_getinfo($client, CURLINFO_HTTP_CODE);
33
34         //Make sure the HTTP request succeeded.
35         if($httpcode == 200){
36
37             //Initialize the JSON object.
38             $json_response = json_decode($http_response, true);
39
40             //Make sure the JSON object was initialized.
41             if($json_response){
42
43                 //Make sure the internal response code was successful.
44                 if($json_response['errno'] === 0){
45
46                     //Retrieve the specified field data.
47                     $field_data = $json_response[$field];
48
49                     //Make sure the data exists and return it.
50                     if($data){
51                         return $field_data;
52                     }
53                     else{
54                         echo 'Error: No data available.';
55                     }
56                 }
57             }
58         }
59     }
60 }
```

```
58         echo 'Error: ' . $response_arr['message'];
59     }
60 }
61 else{
62     echo 'Error: Response could not be parsed.';
63 }
64 }
65 else{
66     echo 'Error: Request failed.';
67 }
68 }
69 else{
70     echo 'Error: No response.';
71     return false;
72 }
73 }
74
75 ?>
```


PTEMPLATE – PACKAGE TEMPLATE

Package template functionality.

9.1 Resources

9.1.1 PtemplateUUID

class PtemplateUUID ()

This resource represents a specific package template.

Resource URI:

`/rest/hosting/ptemplate/<UUID>/`

GET ()

Retrieve the specified package template by submitting an *HTTP GET* request. The *UUID* in the resource uri should be the uuid of the target package template.

Note: This resource requires *read* permission on both the package template and the *elastic_hosting* extension module when using the *HTTP GET* method.

Return type `ptemplate.RepPtemplateUUID`

9.1.2 PtemplateList

class PtemplateList ()

This resource represents a list of package templates.

Resource URI:

`/rest/hosting/ptemplate/list/`

GET ()

Retrieve a list of package templates by submitting an *HTTP GET* request.

Note: This resource requires *read* permission on the *elastic_hosting* extension module.

Return type `ptemplate.RepPtemplateList`

9.2 Representations

9.2.1 RepPtemplateUUID

class RepPtemplateUUID ()

This is a representation of a package template resource.

Representation:

```
{"uuid": ptemplate.uuid,  
 "name": ptemplate.name,  
 "storage": ptemplate.storage,  
 "description": ptemplate.description,  
 "os": ptemplate.os}
```

9.2.2 RepPtemplateList

class RepPtemplateList ()

This is a representation of a list of package templates. This is a list of `ptemplate.RepPtemplateUUID` representations.

Representation:

```
[ptemplate1, ptemplate2]
```

9.3 Examples

9.3.1 PHP

Here is an example of how to list all available package templates.

```
1 <?php  
2  
3 //Make an API call to retrieve all package templates.  
4 $packages = ecp_api_get($client, $base_url.'/rest/hosting/ptemplate/list/', 'packages');  
5  
6 //Make sure there was a response.  
7 if($packages) {  
8  
9     //Display all returned package data.  
10    echo '<h1>Software templates:</h1>';  
11    echo '<ol>';  
12    foreach ($packages as $package) {  
13        echo "<li>$package[name]: $package[description]</li>";  
14    }  
15    echo '</ol>';  
16 }  
17 else {  
18    echo 'No software templates available!';  
19 }  
20  
21 //Basic function to make an API request.
```

```
22 function api_request($client, $url, $field){
23
24     //Make the request and initialize the HTTP response.
25     curl_setopt($client, CURLOPT_URL, $url);
26     $http_response = curl_exec($client);
27
28     //Make sure the API responded.
29     if($http_response){
30
31         //Initialize the response code.
32         $http_response_code = curl_getinfo($client, CURLINFO_HTTP_CODE);
33
34         //Make sure the HTTP request succeeded.
35         if($httpcode == 200){
36
37             //Initialize the JSON object.
38             $json_response = json_decode($http_response, true);
39
40             //Make sure the JSON object was initialized.
41             if($json_response){
42
43                 //Make sure the internal response code was successful.
44                 if($json_response['errno'] === 0){
45
46                     //Retrieve the specified field data.
47                     $field_data = $json_response[$field];
48
49                     //Make sure the data exists and return it.
50                     if($data){
51                         return $field_data;
52                     }
53                     else{
54                         echo 'Error: No data available.';
55                     }
56                 }
57                 else{
58                     echo 'Error: '.$response_arr['message'];
59                 }
60             }
61             else{
62                 echo 'Error: Response could not be parsed.';
63             }
64         }
65         else{
66             echo 'Error: Request failed.';
67         }
68     }
69     else{
70         echo 'Error: No response.';
71         return false;
72     }
73 }
74
75 ?>
```

QUOTA – RESOURCE QUOTA

Virtual machine resource quota functionality.

10.1 Resources

10.1.1 Quota

class `Quota` ()

This resource represents a method in which to create a new quota.

Resource URI:

`/rest/hosting/quota/`

PUT (*name, max_vms, max_cpus, max_memory, max_storage*)

Create a new quota by submitting an *HTTP PUT* request.

Note: This resource requires *create* permission on the *elastic_hosting* extension module.

Parameters

- *name* – The name of the new quota.
- *max_vms* – The maximum number of virtual machines allowed by the quota.
- *max_cpus* – The maximum number of virtual cpus allowed by the quota.
- *max_memory* – The maximum amount of memory allowed by the quota.
- *max_storage* – The maximum amount storage allowed by the quota.

Return type `quota.RepQuota`

10.1.2 QuotaUUID

class `QuotaUUID` ()

This resource represents a specific virtual machine resource quota.

Resource URI:

`/rest/hosting/quota/<UUID>/`

GET ()

Retrieve the specified virtual machine resource quota by submitting an *HTTP GET* request. The *UUID* in the resource URI should be the UUID of the target quota.

Note: This resource requires *read* permission on both the quota and the *elastic_hosting* extension module when using the *HTTP GET* method.

Return type `quota.RepQuotaUUID`

POST (*name, max_vms, max_cpus, max_memory, max_storage*)

Update the specified virtual machine resource quota by submitting an *HTTP POST* request. The *UUID* in the resource uri should be the uuid of the target quota.

Note: This resource requires *update* permission on both the quota and the *elastic_hosting* extension module when using the *HTTP POST* method.

Parameters

- *name* – The name of the new quota.
- *max_vms* – The maximum number of virtual machines allowed by the quota.
- *max_cpus* – The maximum number of virtual cpus allowed by the quota.
- *max_memory* – The maximum amount of memory allowed by the quota.
- *max_storage* – The maximum amount storage allowed by the quota.

Return type `quota.RepQuotaUUID`

DELETE ()

Delete the specified virtual machine resource quota by submitting an *HTTP DELETE* request. The *UUID* in the resource URI should be the UUID of the target quota.

Note: This resource requires *delete* permission on both the quota and the *elastic_hosting* extension module when using the *HTTP DELETE* method.

10.1.3 QuotaGroupUUID

class QuotaGroupUUID ()

This resource represents a specific group quota.

Resource URI:

```
/rest/hosting/quota/group/<UUID>/
```

GET ()

Retrieve the specified virtual machine resource quota by submitting an *HTTP GET* request. The *UUID* in the resource URI should be the UUID of the target group.

Note: This resource requires *read* permission on both the quota and the *elastic_hosting* extension module when using the *HTTP GET* method.

Return type `quota.RepQuotaUUID`

POST (*quota*)

Update the specified group by submitting an *HTTP POST* request. The *UUID* in the resource URI should be the UUID of the target group.

Note: This resource requires *update* permission on both the quota and the *elastic_hosting* extension module when using the *HTTP POST* method.

Parameter *quota* – The new quota for the target group.

Return type `quota.RepQuotaUUID`

10.1.4 QuotaList

class `QuotaList` ()

This resource represents a list of virtual machine resource quotas.

Resource URI:

```
/rest/hosting/quota/list/
```

GET ()

Retrieve a list of virtual machine resource quotas by submitting an *HTTP GET* request.

Note: This resource requires *read* permission on the *elastic_hosting* extension module and on each virtual machine resource quota that is found.

Return type `quota.RepQuotaList`

10.2 Representations

10.2.1 RepQuota

class `RepQuota` ()

This is a representation of a new quota that is been created.

Representation:

```
{"errno" response.errno,
 "message": response.message,
 "quota": quota}
```

10.2.2 RepQuotaUUID

class `RepQuotaUUID` ()

This is a representation of a quota resource.

Representation:

```
{"errno": response.errno,
 "message": response.message,
 "quota": {"uuid": quota.uuid,
           "name": quota.name,
           "max_vms": quota.vms,
           "max_cpus": quota.cpus,
           "max_memory": quota.max_memory,
           "max_storage": quota.max_storage}}
```

10.2.3 RepQuotaList

class RepQuotaList ()

This is a representation of a list of virtual machine resource quotas. This is a list of `quota.RepQuotaUUID` representations.

Representation

```
{"errno": response.errno,  
 "message": response.message,  
 "quotas": [quota1, quota2]}
```

TAG – TAGGING

Virtual machine tagging functionality.

11.1 Resources

11.1.1 Tag

class Tag ()

This resource represents a method in which to create a new set of tags for a specific VM.

Resource URI::

```
/rest/hosting/vm/<UUID>/tag/
```

PUT (*tag*)

Create a new set of tags for a virtual machine by submitting an *HTTP PUT* request. The *tag* parameter is a comma-separated list of tag names. The *UUID* in the resource URI should be the *UUID* of the target virtual machine.

Note: This resource requires *create* permission on the *elastic_hosting* extension module.

Parameter *tag* – The comma-separated list of tags to create.

Return type `tag.RepTag`

See Also:

`vm`

11.1.2 TagName

class TagName ()

This resource represents a specific tag for a target virtual machine referenced by tag name.

Resource URI::

```
/rest/hosting/vm/<UUID>/tag/<NAME>/
```

DELETE ()

Delete the specified tag from the target virtual machine. The *NAME* in the resource URI should be the name of the tag to delete. The *UUID* in the resource URI should be the *UUID* of the target virtual machine.

Note: This resource requires *create* permission on the *elastic_hosting* extension module.

Return type `tag.RepTagList`

See Also:

`vm`

11.1.3 TagList

class TagList ()

This class represents a list of tags. Depending on the resource URI, this can either be a list of all VM tags, or a list of tags belonging to a particular virtual machine.

Resource URI::

```
/rest/hosting/vm/tag/list/  
/rest/hosting/vm/<UUID>/tag/list/
```

GET ()

Retrieve a list of virtual machine tags.

Note: This resource requires *read* permission on the *elastic_hosting* extension module.

Return type `tag.RepTagList`

11.2 Representations

11.2.1 RepTag

class RepTag ()

This is a representation of a set of tags that have just been created.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general message generated by the resource.

tags

This attribute represents the list of tags that have been created.

Representation:

```
{"errno":response.errno,  
 "message":response.message,  
 "tags": [tag1, tag2]}
```

11.2.2 RepTagList

class RepTagList ()

This is a representation of a set of tags.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general message generated by the resource.

tags

This attribute represents the list of tag name strings.

Representation:

```
{"errno":response.errno,  
 "message":response.message,  
 "tags": [tag1, tag2]}
```


USAGE – VIRTUAL MACHINE RESOURCE USAGE

Virtual machine resource usage functionality.

12.1 Resources

12.1.1 Usage

class Usage ()

This resource represents virtual machine resource usage data.

Resource URI:

`/rest/hosting/usage/`

GET ()

Retrieve the usage data by submitting an *HTTP GET* request.

Note: This resource requires *read* permission on the *elastic_hosting* extension module.

Return type `usage.RepUsage`

12.1.2 UsageUUID

class UsageUUID ()

This resource represents virtual machine resource usage data for a particular virtual machine.

Resource URI:

`/rest/hosting/usage/<UUID>/`

GET ()

Retrieve the usage data by submitting an *HTTP GET* request. The *UUID* in the resource path should be the *uuid* of the target virtual machine.

Note: This resource requires *read* permission both on the *elastic_hosting* extension module and on the target virtual machine.

Return type `usage.RepUsage`

12.2 Representations

12.2.1 RepUsage

class RepUsage ()

This is a representation of the virtual machine resource usage data.

Representation:

```
{"vms": {"used": usage.used_vms, "available": usage.available_cpus},
"cpus": {"used": usage.used_cpus, "available": usage.available_cpus},
"memory": {"used": usage.used_memory, "available": usage.available_memory}, \
"storage": {"used": usage.used_storage, "available": usage.available_storage}}
```

USER – USER

User functionality.

13.1 Resources

13.1.1 User

class `User` ()

This resource represents a method in which to create a new user.

Resource URI:

```
/rest/hosting/user/
```

PUT (*username, displayname, password, email*)

Create a new user by submitting an *HTTP PUT* request.

Note: This resource requires *create* permission on the *elastic_hosting* extension module.

Parameters

- *name* – New user name.
- *displayname* – New user display name.
- *password* – New user password.
- *email* – New user email.

Return type `user.RepUser`

13.1.2 UserUUID

class `UserUUID` ()

This resource represents a specific user.

Resource URI:

```
/rest/hosting/user/<UUID>/
```

GET ()

Retrieve the specified user by submitting an *HTTP GET* request. The *UUID* in the resource URI should be the *UUID* of the target user.

Note: This resource requires *read* permission on both the user and the *elastic_hosting* extension module when using the *HTTP GET* method.

Return type `user.RepUserUUID`

POST (*username, displayname, password, email*)

Update the specified user by submitting an *HTTP POST* request. The *UUID* in the resource URI should be the *UUID* of the target user.

Note: This resource requires *update* permission on both the user and the *elastic_hosting* extension module when using the *HTTP POST* method.

Parameters

- *username* – The username to update the user with.
- *displayname* – The display name to update the user with.
- *password* – The password to update the user with.
- *email* – The email address to update the user with.

Return type `user.RepUserUUID`

13.1.3 UserList

class `UserList` ()

This resource represents a list of users.

Resource URI:

`/rest/hosting/user/list/`

GET ()

Retrieve a list of users by submitting an *HTTP GET* request.

Note: This resource requires *read* permission on the *elastic_hosting* extension module and on each user that is found.

Return type `user.RepUserList`

13.2 Representations

13.2.1 RepUser

class `RepUser` ()

This is a representation of a user resource.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general message generated by the resource.

user

This attribute represents the newly created user object and is a `user.RepUserUUID` representation.

Representation:

```
{ "errno": response.errno,
  "message": response.message,
  "user": user_obj }
```

13.2.2 RepUserUUID

class RepUserUUID ()

This is a representation of a user resource.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general message generated by the resource.

uuid

This attribute represents the UUID of the user.

user_name

This attribute represents the username of the user.

display_name

This attribute represents the display name of the user.

password

This attribute represents the password of the user.

email

This attribute represents the email address of the user.

Representation:

```
{ "errno": response.errno,
  "message": response.message,
  "user": { "uuid": user.uuid,
            "user_name": user.user_name,
            "display_name": user.display_name,
            "password": user.password,
            "email": user.email } }
```

13.2.3 RepUserList

class RepUserList ()

This is a representation of a list of users. This is a list of `user.RepUserUUID` representations.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general message generated by the resource.

users

This attribute represents the list of user representations.

Representation

```
{"errno": response.errno,  
 "message": response.message,  
 "users": [user1, user2]}
```

UTIL – UTILITIES

Utility functionality.

14.1 Resources

14.1.1 ErrCodes

class `ErrCodes` ()

This resource represents the potential error codes that may be returned by any given resource.

Resource URI:

`/rest/hosting/errcodes/`

GET ()

Return the available error codes.

Note: This resource requires *read* permission on the *elastic_hosting* extension module when using the *HTTP GET* method.

Return type `util.RepErrCodes`

14.1.2 I18N

class `I18N` ()

This resource represents the language labels used in the user interface.

Resource URI:

`/rest/hosting/i18n/`

GET (*locale*)

Return the language labels used in the user interface by submitting an *HTTP GET* request.

Note: This resource requires *read* permission on the *elastic_hosting* extension module when using the *HTTP GET* method.

Parameter *locale* – The locale of the translations to return.

Return type `util.RepI18N`

14.1.3 IconMap

class IconMap ()

This resource represents the icons for operation systems and applications used in the user interface.

Resource URI:

```
/rest/hosting/iconmap/
```

GET ()

Return the icons used in the user interface.

Note: This resource requires *read* permission on the *elastic_hosting* extension module when using the *HTTP GET* method.

Return type `util.RepIconMap`

14.2 Representations

14.2.1 RepErrCodes

class RepErrCodes ()

This is a representation of the available API error codes.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general error message generated by the resource.

errcodes

This attribute represents the errcode dictionary.

Representation:

```
{"errno": response.errno,  
 "message": response.message,  
 "errcodes": {"err_label": err.number}}
```

14.2.2 Repl18N

class Repl18N ()

This is a representation of language labels used in the user interface.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general error message generated by the resource.

i18n

This attribute represents the dictionary of language translations.

Representation:

```
{"errno": response.errno,  
 "message": response.message,  
 "i18n": {"current": i18n.current,  
         "supported": i18n.supported,  
         "i18n_label": i18n.translation}}
```

14.2.3 ReplconMap

class RepIconMap ()

This is a representation of the icons used in the user interface.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general error message generated by the resource.

icons

This attribute represents the dictionary of operating system icons.

Representation:

```
{"errno": response.errno,  
 "message": response.message,  
 "icons": {"base_path": "/path/",  
          "os_label": {"small": os.small,  
                      "medium": os.medium}}}
```


VM – VIRTUAL MACHINE

Virtual machine functionality.

15.1 Resources

15.1.1 Vm

class Vm()

This resource represents a method in which to provision a new virtual machine.

Resource URI:

`/rest/hosting/vm/`

PUT (*name, package, hardware, disk, network_uuid*)

Provision a new virtual machine by submitting an *HTTP PUT* request.

Note: This resource requires *create* permission on the *elastic_hosting* extension module.

Parameters

- *name* – New virtual machine name.
- *package* – New virtual machine package.
- *hardware* – New virtual machine hardware profile.
- *disk* – Additional hard disk for the virtual machine.
- *network_uuid* – Network associated with the new virtual machine.

Return type `vm.RepVm`

See Also:

`ptemplate htemplate dtemplate network`

15.1.2 VmUUID

class VmUUID()

This resource represents a specific virtual machine.

Resource URI:

/rest/hosting/vm/<UUID>/

GET ()

Retrieve the specified virtual machine by submitting an *HTTP GET* request. The *UUID* in the resource URI should be the UUID of the target virtual machine.

Note: This resource requires *read* permission on both the virtual machine and the *elastic_hosting* extension module when using the *HTTP GET* method.

Return type `vm.RepVmUUID`

POST (*action, name, hardware_profile_uuid*)

Update the specified virtual machine by submitting an *HTTP POST* request. The *UUID* in the resource URI should be the UUID of the target virtual machine.

Note: This resource requires *update* permission on both the virtual machine and the *elastic_hosting* extension module when using the *HTTP POST* method.

Parameters

- *action* – The action to perform on the virtual machine. The following actions are supported - *start, stop, pause, resume, delete, and save.*
- *name* – The name to update the virtual machine with.
- *hardware_profile_uuid* – The hardware profile to associate with the virtual machine.

Return type `vm.RepVmUUID`

15.1.3 VmList

class VmList ()

This resource represents a list of virtual machines.

Resource URI:

/rest/hosting/vm/list/

GET ()

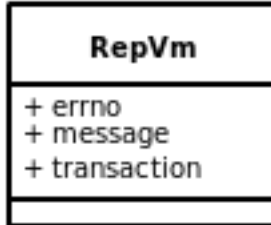
Retrieve a list of virtual machines by submitting an *HTTP GET* request.

Note: This resource requires *read* permission on the *elastic_hosting* extension module and on each virtual machine that is found.

Return type `vm.RepVmList`

15.2 Representations

15.2.1 RepVm



class RepVm()

This is a representation of a virtual machine currently being provisioned.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general message generated by the resource.

transaction

This attribute represents the UUID of the transaction that is executing the provision job.

Representation:

```
{ "errno": response.errno,
  "message": response.message,
  "transaction": transaction.uuid }
```

15.2.2 RepVmUUID

class RepVmUUID()

This is a representation of a virtual machine resource.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general message generated by the resource.

uuid

This attribute represents the UUID of the virtual machine.

name

This attribute represents the name of the virtual machine.

state

This attribute represents the current state of the virtual machine.

os

This attribute represents the operating system of the virtual machine.

hardware_profile_uuid

This attribute represents the UUID of the hardware profile this virtual machine is using.

vnc_ip_address

This attribute represents the VNC IP address of the virtual machine.

vnc_port

This attribute represents the VNC port of the virtual machine.

vnc_password

This attribute represents the VNC password of the virtual machine.

vnc_enabled

This attribute represents whether or not VNC is enabled for this virtual machine.

interfaces

This attribute represents the list of network interfaces belonging to this virtual machine.

Representation:

```
{"errno": response.errno,  
"message": response.message,  
"vm": {"uuid": vm.uuid,  
       "name": vm.name,  
       "state": vm.state,  
       "os": vm.os,  
       "hardware_profile_uuid": vm.hardware_profile,  
       "vnc_ip_address": vm.vnc_ip,  
       "vnc_port": vm.vnc_port,  
       "vnc_password": vm.vnc_password,  
       "vnc_enabled": vm.vnc_enabled,  
       "interfaces": [interface_obj1, interface_obj2]}}
```

15.2.3 RepVmList

class RepVmList ()

This is a representation of a list of virtual machines. This is a list of `vm.RepVmUUID` representations.

errno

This attribute represents the general error number generated by the resource.

message

This attribute represents the general message generated by the resource.

vms

This attribute represents the list of virtual machine representations.

Representation

```
{"errno": response.errno,  
"message": response.message,  
"vms": [vm1, vm2]}
```

See Also:

[vm.RepVmUUID](#)

INDICES AND TABLES

- *Index*
- *Module Index*
- *Search Page*

MODULE INDEX

B

billing, 7

C

clone, 9

D

dtemplate, 11

G

group, 17

H

htemplate, 21

L

log, 27

N

network, 31

P

ptemplate, 37

Q

quota, 41

T

tag, 45

U

usage, 49

user, 51

util, 55

V

vm, 59

INDEX

B

Billing (class in billing), 7
billing (module), 7

C

Clone (class in clone), 9
clone (module), 9

D

DELETE() (htemplate.HtemplateUUID method), 22
DELETE() (network.NetworkUUID method), 32
DELETE() (quota.QuotaUUID method), 42
DELETE() (tag.TagName method), 45
display_name (group.RepGroupUUID.group attribute),
19
display_name (user.RepUserUUID.user attribute), 53
dtemplate (module), 11
DtemplateList (class in dtemplate), 11
DtemplateUUID (class in dtemplate), 11

E

email (user.RepUserUUID.user attribute), 53
ErrCodes (class in util), 55
errcodes (util.RepErrCodes attribute), 56
errno (clone.RepClone attribute), 9
errno (dtemplate.RepDtemplateList attribute), 13
errno (dtemplate.RepDtemplateUUID attribute), 12
errno (group.RepGroup attribute), 18
errno (group.RepGroupList attribute), 19
errno (group.RepGroupUUID attribute), 19
errno (log.RepLogList attribute), 28
errno (network.RepNetwork attribute), 32
errno (network.RepNetworkList attribute), 33
errno (network.RepNetworkUUID attribute), 33
errno (tag.RepTag attribute), 46
errno (tag.RepTagList attribute), 46
errno (user.RepUser attribute), 52
errno (user.RepUserList attribute), 53
errno (user.RepUserUUID attribute), 53
errno (util.RepErrCodes attribute), 56
errno (util.RepI18N attribute), 56

errno (util.RepIconMap attribute), 57
errno (vm.RepVm attribute), 61
errno (vm.RepVmList attribute), 62
errno (vm.RepVmUUID attribute), 61
error (log.RepLogUUID attribute), 28

G

GET() (billing.Billing method), 7
GET() (dtemplate.DtemplateList method), 11
GET() (dtemplate.DtemplateUUID method), 11
GET() (group.GroupList method), 18
GET() (group.GroupUUID method), 17
GET() (htemplate.HtemplateList method), 22
GET() (htemplate.HtemplateUUID method), 21
GET() (log.LogList method), 27
GET() (log.LogUUID method), 27
GET() (network.NetworkList method), 32
GET() (network.NetworkUUID method), 31
GET() (ptemplate.PtemplateList method), 37
GET() (ptemplate.PtemplateUUID method), 37
GET() (quota.QuotaGroupUUID method), 42
GET() (quota.QuotaList method), 43
GET() (quota.QuotaUUID method), 41
GET() (tag.TagList method), 46
GET() (usage.Usage method), 49
GET() (usage.UsageUUID method), 49
GET() (user.UserList method), 52
GET() (user.UserUUID method), 51
GET() (util.ErrCodes method), 55
GET() (util.I18N method), 55
GET() (util.IconMap method), 56
GET() (vm.VmList method), 60
GET() (vm.VmUUID method), 60
Group (class in group), 17
group (group.RepGroup attribute), 18
group (module), 17
group_name (group.RepGroupUUID.group attribute), 19
GroupList (class in group), 18
groups (group.RepGroupList attribute), 19
GroupUUID (class in group), 17

H

hardware_profile_uuid (vm.RepVmUUID.vm attribute), 61

Htemplate (class in htemplate), 21

htemplate (module), 21

HtemplateList (class in htemplate), 22

HtemplateUUID (class in htemplate), 21

I

I18N (class in util), 55

i18n (util.RepI18N attribute), 56

IconMap (class in util), 56

icons (util.RepIconMap attribute), 57

interfaces (vm.RepVmUUID.vm attribute), 62

L

log (module), 27

LogList (class in log), 27

logs (log.RepLogList attribute), 28

LogUUID (class in log), 27

M

machine_id (clone.RepClone attribute), 9

message (clone.RepClone attribute), 9

message (dtemplate.RepDtemplateList attribute), 13

message (dtemplate.RepDtemplateUUID attribute), 12

message (group.RepGroup attribute), 18

message (group.RepGroupList attribute), 19

message (group.RepGroupUUID attribute), 19

message (log.RepLogList attribute), 28

message (log.RepLogUUID attribute), 28

message (network.RepNetwork attribute), 32

message (network.RepNetworkList attribute), 33

message (network.RepNetworkUUID attribute), 33

message (tag.RepTag attribute), 46

message (tag.RepTagList attribute), 46

message (user.RepUser attribute), 52

message (user.RepUserList attribute), 53

message (user.RepUserUUID attribute), 53

message (util.RepErrCodes attribute), 56

message (util.RepI18N attribute), 56

message (util.RepIconMap attribute), 57

message (vm.RepVm attribute), 61

message (vm.RepVmList attribute), 62

message (vm.RepVmUUID attribute), 61

N

name (network.RepNetworkUUID.network attribute), 33

name (vm.RepVmUUID.vm attribute), 61

Network (class in network), 31

network (module), 31

network (network.RepNetwork attribute), 32

NetworkList (class in network), 32

networks (network.RepNetworkList attribute), 33

NetworkUUID (class in network), 31

O

os (vm.RepVmUUID.vm attribute), 61

P

password (user.RepUserUUID.user attribute), 53

POST() (clone.Clone method), 9

POST() (group.GroupUUID method), 18

POST() (htemplate.HtemplateUUID method), 22

POST() (network.NetworkUUID method), 31

POST() (quota.QuotaGroupUUID method), 42

POST() (quota.QuotaUUID method), 42

POST() (user.UserUUID method), 52

POST() (vm.VmUUID method), 60

ptemplate (module), 37

PtemplateList (class in ptemplate), 37

PtemplateUUID (class in ptemplate), 37

PUT() (group.Group method), 17

PUT() (htemplate.Htemplate method), 21

PUT() (network.Network method), 31

PUT() (quota.Quota method), 41

PUT() (tag.Tag method), 45

PUT() (user.User method), 51

PUT() (vm.Vm method), 59

Q

Quota (class in quota), 41

quota (module), 41

QuotaGroupUUID (class in quota), 42

QuotaList (class in quota), 43

QuotaUUID (class in quota), 41

R

RepBilling (class in billing), 7

RepClone (class in clone), 9

RepDtemplateList (class in dtemplate), 13

RepDtemplateUUID (class in dtemplate), 12

RepErrCodes (class in util), 56

RepGroup (class in group), 18

RepGroupList (class in group), 19

RepGroupUUID (class in group), 19

RepHtemplate (class in htemplate), 22

RepHtemplateList (class in htemplate), 24

RepHtemplateUUID (class in htemplate), 23

RepI18N (class in util), 56

RepIconMap (class in util), 57

RepLogList (class in log), 28

RepLogUUID (class in log), 28

RepNetwork (class in network), 32

RepNetworkList (class in network), 33

RepNetworkUUID (class in network), 33

RepPtemplateList (class in ptemplate), 38
 RepPtemplateUUID (class in ptemplate), 38
 RepQuota (class in quota), 43
 RepQuotaList (class in quota), 44
 RepQuotaUUID (class in quota), 43
 RepTag (class in tag), 46
 RepTagList (class in tag), 46
 RepUsage (class in usage), 50
 RepUser (class in user), 52
 RepUserList (class in user), 53
 RepUserUUID (class in user), 53
 RepVm (class in vm), 61
 RepVmList (class in vm), 62
 RepVmUUID (class in vm), 61

Vm (class in vm), 59
 vm (module), 59
 VmList (class in vm), 60
 vms (vm.RepVmList attribute), 62
 VmUUID (class in vm), 59
 vnc_enabled (vm.RepVmUUID.vm attribute), 62
 vnc_ip_address (vm.RepVmUUID.vm attribute), 61
 vnc_password (vm.RepVmUUID.vm attribute), 62
 vnc_port (vm.RepVmUUID.vm attribute), 62

S

size (dtemplate.RepDtemplateUUID attribute), 12
 state (vm.RepVmUUID.vm attribute), 61

T

Tag (class in tag), 45
 tag (module), 45
 TagList (class in tag), 46
 TagName (class in tag), 45
 tags (tag.RepTag attribute), 46
 tags (tag.RepTagList attribute), 47
 templates (dtemplate.RepDtemplateList attribute), 13
 timestamp (log.RepLogUUID attribute), 28
 transaction (vm.RepVm attribute), 61
 transactions (log.RepLogList attribute), 28

U

Usage (class in usage), 49
 usage (module), 49
 UsageUUID (class in usage), 49
 User (class in user), 51
 user (log.RepLogUUID attribute), 28
 user (module), 51
 user (user.RepUser attribute), 52
 user_name (user.RepUserUUID.user attribute), 53
 UserList (class in user), 52
 users (user.RepUserList attribute), 53
 UserUUID (class in user), 51
 util (module), 55
 uuid (dtemplate.RepDtemplateUUID attribute), 12
 uuid (group.RepGroupUUID.group attribute), 19
 uuid (network.RepNetworkUUID.network attribute), 33
 uuid (user.RepUserUUID.user attribute), 53
 uuid (vm.RepVmUUID.vm attribute), 61

V

vlan_id (network.RepNetworkUUID.network attribute),
 33