VNS3:ms 1.5.4

API v1

Reference Document v20171009

Table of Contents

Reference Document v20171009

VNS3:ms API Version 1

This is the Cohesive Networks VNS3:ms API Reference. This document provides descriptions, syntax, and examples for each of the API actions for the VNS3:ms version 1.5.4. API examples will show how to access using generic API URIs with curl command examples. All actions available via the VNS3:ms web UI are available using the RESTful API.

Getting help with VNS3:ms
Before accessing the VNS3:ms API, launch a VNS3:ms instance, change the default password and regenerate the VNS3:ms API token via the UI or API. If these actions are not taken functionality will be limited. Additionally this document assumes you have a VNS3 Controller instance launched and running in a security group, network or similar that has the appropriate access rules included to allow the VNS3:ms instance to access the VNS3 controller on TCP port 8000.

See the specific instructions for your cloud setup and instance launch on the Cohesive Networks Product Resources page (http://www.cohesive.net/support/product-resources/).

Please review the VNS3 Support Plans (http://www.cohesive.net/support/support-plans/) and Support Contacts (http://www.cohesive.net/support/support-contacts/) before sending support inquiries. If you need specific help with project planning, POCs, or audits, contact our professional services team via sales@cohesive.net for details.

Reference Document v20171009

Authentication and Authorisation

Authentication and authorisation are used to login to the API system and determine permissions. The approach is to use a user's API key and receive a token. The token will be used for all other API calls.

An API token expires 10 minutes after the last use, or they can be tracked and disabled by the user.

In the examples the following environmental variables may appear:

```
$ export type="Content-Type: application/json"
$ export version="Accept-Version:v1"
$ export token="api-token:095070c0baadf663a6c1d9cc9d0740aab729badda41901333b7fe558501dca02..."
```

(Note: the token will be returned from the POST /api/auth call and will be different from above)

PUT /api/auth/activate

Activate default key

Note: The user's default key is not activated by default. Activation replaces the default, publicly known API key value with a generated version. This can only be retrieved one time.

Parameters:

- username (required) : The username for the owner of the API key
- api_key (required) : Default user API key

Example:

```
$ export api_key="changeme"
$ curl -k -H "$type" -H "$version" -X POST \
       -d '{"username":"admin", "api_key":"'$api_key'"}' https://{server}/api/auth
{
   "response_type" : "success",
   "response" : {
      "message" : "Key activated",
      "name" : "Activated Default Key",
      "api_key" : "2383e1b47349c23bb1818a27a963fa31e80680b74cdee4123e77ee7ef084bd2604831234...",
      "id" : 1
   }
}
```

POST /api/auth

Login and receive an API token

Parameters:

- username (required) : The username for the owner of the API key
- api_key (required) : An API key

Example:

```
$ curl -k -H "$type" -H "$version" -X POST \
       -d '{"username":"admin", "api_key":"real_api_key"}' https://{server}/api/auth
{ "api_token" : "095070c0baadf663a6c1d9cc9d0740aab729badda41901333b7fe558501dca02" }
```

GET /api/auth/api_keys

List active keys

Example:

```
$ curl -k -H "$token" -H "$type" -H "$version" -X GET https://{server}/api/auth/api_keys
{
   "response_type" : "success",
   "response" : [
      {
         "id" : 4,
         "name" : "Activated Default Key",
         "enabled" : true,
         "default_key" : false,
         "active_tokens" : 1,
         "expired_tokens" : 1,
         "last_access_ip" : "172.16.174.141",
         "last_access_time" : "2016-09-02T04:02:09.617Z",
         "created_at" : "2016-08-25T18:23:02.840Z"
      },
      {
         "id" : 10,
         "name" : "foo1",
         "enabled" : true,
         "default_key" : false,
         "active_tokens" : 0,
         "expired_tokens" : 0,
         "created_at" : "2016-08-26T21:11:08.828Z"
      }
   ]
}
```

Note: If the key has not been accessed then fields such as last_access_time and last_access_ip will not appear in the returned values.

POST /api/auth/api_key

Generate new key

Parameters:

- key_name (required) : Name/description of API key

Example:

```
$ curl -k -H "$token" -H "$type" -H "$version" -X POST \
       -d '{"key_name": "blah blah blah blah blah"}' https://{server}/api/auth/api_key
{
   "response_type" : "success",
   "response" : {
      "message" : "Key created",
      "name" : "blah blah blah blah blah",
      "enabled" : true,
      "api_key" : "2383e1b47349c23bb1818a27a963fa31e80680b74cdee4123e77ee7ef084bd26",
      "id" : 3
   }
}
```

PUT /api/auth/api_key/:id

Update existing key

Parameters:

- api_key_id (required) : API Key ID
- key_name (optional) : Name/description of API key
- enabled (optional) : Enable/disable the key

Note: The update is strictly limited -- change either the name of the key or the status. The actual generated key value can not be changed.

Example:

```
$ curl -k -H "$token" -H "$type" -H "$version" -X PUT -d '{"enabled": false}' \
       https://{server}/api/auth/api_key/3
{
   "response_type" : "success",
   "response" : {
      "message" : "Key updated",
      "name" : "blah blah blah blah blah",
      "enabled" : false,
      "id" : 3
```

```
   }
}
```

DELETE /api/auth/api_key

Delete API key

Parameters:

- api_key_id (required) : API Key ID

Example:

```
$ curl -k -H "$token" -H "$type" -H "$version" -X DELETE https://{server}/api/auth/api_key/2
{
   "response_type" : "success",
   "response" : "Deleted key"
}
```

PUT /api/auth/expire_token

Expire (logout) the current token

Parameters: None

Example:

```
$ curl -k -H "$token" -H "$type" -H "$version" -X PUT https://{server}/api/auth/expire_token
{ "response_type" : "success", "response" : { "message" : "Successfully expired token" }}
```

PUT /api/auth/invalidate_tokens

Invalidate all tokens for a specific API key

Parameters:

- api_key_id (required) : API Key ID

Example:

```
$ curl -k -H "$token" -H "$type "-H "$version" -X PUT \
       -d '{"api_key_id": 2}' https://{server}/api/auth/invalidate_tokens
{
   "response_type" : "success",
```

```
   "response" : {
      "message" : "Invalidated 2 tokens",
      "name" : "default",
      "id" : 2,
      "enabled":true
   }
}
```

System

GET /api/system/ping

Return an echo to a ping - does not require authentication

Example:

```
$ curl -kL http://{server}/api/system/ping
{"api_version":"1.0","response":"Alive at 2014-12-07 18:32:45 -0600"}
```

GET /api/system/authenticated_ping

Return an echo to a ping - does require authentication

Example:

```
$ curl -kL http://{server}/api/system/authenticated_ping
{"response_type":"error","response":"401 Unauthorised"}

$ curl -kL http://{server}/api/system/authenticated_ping -H "api-token:foo"
{"api_version":"1.0","response":"Authenticated and alive at 2014-12-07 18:32:45 -0600"}
```

GET /api/system/remote_support

Return current status of remote support: enabled or disabled

Example:

```
$ curl -kL -H "api-token:foo" -X GET http://{server}/api/system/remote_support
{"response_type":"success","response":{"remote_support_enabled":true}}
```

PUT /api/system/remote_support

Enable/disable remote support

Parameters:

- enable (required) : boolean value, 'true' or 'false'

Example:

```
$ curl -kL -H "api-token:foo" -X PUT http://{server}/api/system/remote_support?enable=true
```

```
{"response_type":"success","response":{"remote_support_enabled":true}}
$ curl -kL -H "api-token:foo" -X PUT http://{server}/api/system/remote_support?enable=false
{"response_type":"success","response":{"remote_support_enabled":false}}
```

GET /api/system/remote_support/keypair

Return current status of a support keypair: installed or not

Example:

```
$ curl -kL -H "api-token:foo" -X GET http://{server}/api/system/remote_support/keypair_installed
{"response_type":"success","response":{"keypair_installed":true}}
```

POST /api/system/remote_support/keypair

Generate (or regenerate) and install a remote support keypair

Parameters:

- encrypted_passphrase (required): An encrypted passphrase

Example:

```
$ curl -kL -H "api-token:foo" -X POST -F encrypted_passphrase=@test.key http://
{server}/api/system/remote_support/keypair
{"response_type":"success","response":{"keypair_installed":true,"private_key":"-----BEGIN RSA PRIVATE
KEY-----\nProc-Type: 4,ENCRYPTED\nDEK-Info: AES-128-
CBC,AA14CC660445F52CDDFD523701B6AB01\n\ndD4Q2lG6HlZ/s4KWk9zbTzNnE1K3lp74fXhP5JFFEkoFHFEBG+Lrx5K7jrvRjZ
CI\n...\n-----END RSA PRIVATE KEY-----\n"}}
```

DELETE /api/system/remote_support/keypair

Revoke a keypair (if installed)

Example:

```
$ curl -kL -H "api-token:foo" -X DELETE http://{server}/api/system/remote_support/keypair
{"response_type":"success","response":{"keypair_installed":false}}
```

GET /api/system/status

Return current status of system as a whole

Example:

```
$ curl -kL -H "api-token:foo" -X GET http://{server}/api/system/status
{"response_type":"success","response":{"system_disk":
{"type":"xfs","block_size":4096,"free_blocks":823372,"available_blocks":823372,"total_blocks":5239808}
,"data_disk":
{"type":"xfs","block_size":4096,"free_blocks":21460973,"available_blocks":21460973,"total_blocks":2620
1600},"cpus":[{"num":0,"user":23465,"system":10389,"nice":0,"idle":1518408},
{"num":1,"user":24382,"system":10003,"nice":0,"idle":1519482}],"load_average":
{"one_minute":0.21,"five_minutes":0.11,"fifteen_minutes":0.06},"memory":
{"pagesize":4096,"wired":39417,"active":342564,"inactive":15802,"free":113162,"pageins":7771082,"pageo
uts":8501220},"boot_time":"2015-07-16T12:04:24.011Z","system_time":"2015-07-
16T16:24:33.741Z","uptime":15609.729933617,"virtual_networks":11,"vns3_topologies":13,"vns3_controller
s":36,"cloud_vlans":2,"cloud_vlan_components":6,"ha_snapshot_file_count":6,"configuration_snapshot_fil
e_count":98,"configuration_snapshot_failed_count":0,"system_backup_file_count":6,"system_snapshots_bac
kup_file_count":1}}
```

GET /api/system/credential_types

Return list of credential types

Example:

```
$ curl -kL -H "api-token:foo" -X GET http://{server}/api/system/credential_types
{"response_type":"success","response":[{"name":"EC2","code":"ec2"},
{"name":"Azure","code":"azure","description":"Azure credentials"}]}
```

GET /api/system/credential_types/:code

Return list of credential type details

Example:

```
$ curl -kL -H "api-token:foo" -X GET http://{server}/api/system/credential_types/ec2
{"name":"EC2","code":"ec2","fields":[{"name":"Account ID","description":"AWS Account ID (optional -
VNS3:ms will attempt to fill if left blank)","required":false,"type":"STRING"},{"name":"Access
Key","description":"AWS Access key","required":true,"type":"STRING"},{"name":"Secret
Key","description":"AWS Secret key","required":true,"type":"PASSWORD"},
{"name":"GovCloud","description":"These creds are for use in EC2's
GovCloud","required":true,"type":"BOOLEAN"}]}
```

GET /api/system/ntp_hosts

Return list of NTP hosts

Example:

```
$ curl -kL -H "api-token:foo" -X GET http://{server}/api/system/ntp_hosts
{"response_type":"success","response":
{"0":"0.ubuntu.pool.ntp.org","1":"1.ubuntu.pool.ntp.org","2":"2.ubuntu.pool.ntp.org","3":"3.ubuntu.pool.ntp.org","4":"ntp.ubuntu.com"}}
```

POST /api/system/ntp_hosts

Add an NTP host to the list

Parameters:

- host (required) : New hostname to add to the list

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X POST -d
'{"host":"99.ubuntu.pool.ntp.org"}' http://{server}/api/system/ntp_hosts
{"response_type":"success","response":{"5":"99.ubuntu.pool.ntp.org"}}
```

DELETE /api/system/ntp_hosts/:id

Delete specific NTP host

Parameters:

- id (required) : Integer ID of host to be deleted

Example:

```
$ curl -kL -H "api-token:foo" -X DELETE http://{server}/api/system/ntp_hosts/1
{"response_type":"success","response":"Deleted NTP host 1.ubuntu.pool.ntp.org"}
```

PUT /api/system/ssl/keypair

Upload new SSL cert and key pair

Parameters:

- cert (required) : cert String

•key (required) : key String

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X PUT -d '{"cert":"...",
"key":"..."}' http://{server}/api/system/ssl/keypair
{"response_type":"success","response":"Valid key/cert files uploaded"}

$ curl -kL -H "api-token:foo" -X PUT -F cert=@test.pem -F key=@test.key http://
{server}/api/system/ssl/keypair
{"response_type":"success","response":"Valid key/cert files uploaded"}
```

PUT /api/system/ssl/install

Install new SSL cert/key pair. Files must first be uploaded and verified via the PUT /api/system/ssl/keypair call.

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X PUT http://
{server}/api/system/ssl/install
{"response_type":"success","response":{"status":"SSL Cert installation queued",
"uuid":"12345678901234567890"}}
```

Notes: A job UUID is returned so the status can be checked once the system has restarted. The job status can be checked via a call to GET /api/system/jobs Successful installation of a new SSL cert/key pair requires a reload of the webserver configuration which may lead to a brief period of slower response time until the system has re-cached its resources.

DELETE /api/system/ssl

Revoke existing SSL cert/key pair and generate a new self-signed pair

Example:

```
$ curl -skL -H "api-token:foo" -H "Content-Type: application/json" -X DELETE http://
{server}/api/system/ssl
{"response_type":"success","response":{"status":"SSL Cert revocation queued",
"uuid":"12345678901234567891"}}
```

Notes: A job UUID is returned so the status can be checked.

GET /api/system/controller_report

Get the controller report snapshot, either at the time of request or the saved snapshot for a given date

Parameters:

- report_date (optional) : string in standard YYYY-MM-DD format

Example:

```
$ curl -kL -H "api-token:foo" -X GET http://{server}/api/system/controller_report
{"response_type":"success","response":{"snapshot_date":"2016-02-05","snapshot:
[{"controller_id":1,"controller_name":"test1","controller_version":"3.5.0.00-
20140430","active":false,"created_date":"2016-01-01T12:01:01-
06:00","topology_id":1,"topology_name":"Topo1","virtual_network_id":1,"virtual_network_name":"VN1"},
{"controller_id":2,"controller_name":"test2","controller_version":"3.6.0.0","active":true,"created_dat
e":"2016-01-01T12:01:02-
06:00","topology_id":1,"topology_name":"Topo1","virtual_network_id":1,"virtual_network_name":"VN1","ha
_backup_state":false},{"controller_id":3,"controller_name":"test3","controller_version":"3.5.0.00-
20140430","active":true,"created_date":"2016-02-04T12:01:03-
06:00","topology_id":1,"topology_name":"Topo1","virtual_network_id":1,"virtual_network_name":"VN1","ha
_backup_state":true}]}}

$ curl -kL -H "api-token:foo" http://{server}/api/system/controller_report?report_date=2016-02-01
{"response_type":"success","response":{"snapshot_date":"2016-02-01","snapshot:
[{"controller_id":1,"controller_name":"test1","controller_version":"3.5.0.00-
20140430","active":false,"created_date":"2016-01-01T12:01:01-
06:00","topology_id":1,"topology_name":"Topo1","virtual_network_id":1,"virtual_network_name":"VN1"},
{"controller_id":2,"controller_name":"test2","controller_version":"3.6.0.0","active":true,"created_dat
e":"2016-01-01T12:01:02-
06:00","topology_id":1,"topology_name":"Topo1","virtual_network_id":1,"virtual_network_name":"VN1","ha
_backup_state":false}]}}
```

Notes: Making the request with no date will return the current information. This may or may not match the values in the daily snapshot depending on when the snapshot is taken relative to the request for the report. The snapshot returned will be an empty array if a date is requested but no snapshot exists.

```
{"response_type":"success","response":{"snapshot_date":"2016-02-01","snapshot":[]}}
```

GET /api/system/jobs/:uuid

Get the status of the job identified by a UUID

Parameters:

- uuid (required) : string UUID returned from call to function that queues

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X GET http://
{server}/api/system/ssl/install/12345678901234567890
{"response_type":"success","response":{"uuid":"12345678901234567890","status":"Completed"}}
```

Virtual Networks

GET /api/virtual_networks

Get general list of Virtual Networks

Parameters:

- virtual_network_id (optional) : ID of a particular virtual network

Example:

```
$ curl -skL -H "api-token:foo" http://{server}/api/virtual_networks
{"response":[{"id":1,"name":"Virtual Network 1","description":"Virtual Network #1","created_at":"2014-
09-16T20:21:19.000Z"},{"id":2,"name":"Demo Virtual Network","description":"Blah blah
blah","created_at":"2014-09-16T20:21:20.000Z"}]}
```

Notes: Using the virtual_network_id is supported but we recommend use of the /api/virtual_networks/:id GET method. The database hit is smaller.

POST /api/virtual_networks

Create a new virtual network

Parameters:

- name (required) : Virtual Network name
- description (optional) : Virtual Network description

Example:

```
$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" -X POST -d '{"name":"VN test #1",
"description":"Test VN"}' http://{server}/api/virtual_networks
{"response_type":"success","response":{"msg":"Virtual Network created","id":1}}
```

GET /api/virtual_networks/:id

Get details about a specific Virtual Network based on ID

Example:

```
$ curl -skL -H "api-token:foo" http://{server}/api/virtual_networks/1
{"response":{"id":1,"name":"Virtual Network 1","description":"Virtual Network #1","created_at":"2014-
09-16T20:21:19.000Z"}}
```

PUT /api/virtual_networks/:id

Update specified Virtual Network

Parameters:

- id (required) : Virtual Network ID
- name (optional) : Virtual Network name
- description (optional) : Virtual Network description

Example:

```
$ curl -s -H "api-token:foo" -H "Content-Type:application/json" -d '{"name":"foo",
"description":"bar"}' -X PUT http://{server}/api/virtual_networks/1
{"response_type":"success","response":"Updated Virtual Network"}
```

DELETE /api/virtual_networks/:id

Delete specified Virtual Network

Parameters:

- id (required) : Virtual Network ID

Example:

```
$ curl -H "api-token:foo" -X DELETE http://localhost:43000/api/virtual_networks/1
{"response_type":"success","response":"Deleted Virtual Network 1"}

$ curl -H "api-token:foo" -X DELETE http://localhost:43000/api/virtual_networks/99
{"response_type":"error","response":"Missing Virtual Network with ID 99"}
```

GET /api/virtual_networks/export

Example:

```
$ curl -skL -H "api-token:foo" http://{server}/api/virtual_networks/export > foo2.csv
```

Notes: Creates file as a tab-delimited CSV on success.

POST /api/virtual_networks/import

Parameters:

- import_file (required) : Import filename from local system

Example:

```
$ curl -skL -H "api-token:foo" -X POST -F import_file=@import.csv http://
{server}/api/virtual_networks/import
{"response_type":"success","response":"Import succeeded"}
```

VNS3 Topologies

GET /api/vns3_topologies

Get general list of VNS3 Topologies

Parameters:

- vns3_topology_id (optional) : ID of a particular topology

Example:

```
$ curl -skL -H "api-token:foo" http://{server}/api/vns3_topologies
{"response":[{"id":1,"name":"VNS3 Topology 1","virtual_network_name":"Virtual Network
1","virtual_network_id":"1","vns3_controllers":[{"id":1,"name":"172.16.174.180"},
{"id":2,"name":"172.16.174.181"},{"id":3,"name":"172.16.174.182"}],"description":"VNS3 Topology #1"},
{"id":2,"name":"Demo Topology","virtual_network_name":"Demo Virtual
Network","virtual_network_id":"2","vns3_controllers":[{"id":4,"name":"US-east"},{"id":5,"name":"US-
west"},{"id":6,"name":"EU-west"}],"description":"More blah blah blah"}]}
```

Notes: Using the vns3_topology_id is supported but we recommend use of the /api/vns3_topologies/:id GET method. The database hit is smaller.

GET /api/vns3_topologies/:id

Get details about a specific VNS3 Topology based on ID

Example:

```
$ curl -skL -H "api-token:foo" http://{server}/api/vns3_topologies/1
{"response":{"id":1,"name":"VNS3 Topology 1","virtual_network_name":"Virtual Network
1","virtual_network_id":"1","vns3_controllers":[{"id":1,"name":"172.16.174.180"},
{"id":2,"name":"172.16.174.181"},{"id":3,"name":"172.16.174.182"}],"description":"VNS3 Topology #1"}}
```

POST /api/vns3_topologies

Create a new VNS3 topology

Parameters:

- name (required) : VNS3 Topology name
- virtual_network_id (required) : Parent virtual network ID
- description (optional) : Topology description

Example:

```
$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" -X POST  -d '{"name":"VT test #1",
"virtual_network_id":"1", "description":"Test topology"}' http://{server}/api/vns3_topologies
{"response_type":"success","response":{"msg":"VNS3 Topology created","id":1}}
```

PUT /api/vns3_topologies/:id

Update specified VNS3 topology

Parameters: (All except ID are optional)

- id (required) : Topology ID from VNS3:ms
- name : Topology name
- virtual_network_id : Parent virtual network ID
- description : Topology description

Example:

```
$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" -d '{"name":"foo",
"description":"bar"}' -X PUT http://{server}/api/vns3_topologies/1
{"response_type":"success","response":"Updated VNS3 topology"}
```

DELETE /api/vns3_topologies/:id

Delete specified VNS3 Topology

Parameter:

- id (required) : VNS3 Topology ID

Example:

```
$ curl -H "api-token:foo" -X DELETE http://{server}/api/vns3_topologies/1
{"response_type":"success","response":"Deleted VNS3 Topology 1"}

$ curl -H "api-token:foo" -X DELETE http://{server}/api/vns3_topologies/1
{"response_type":"error","response":"VNS3 Topology with ID 1 does not exist"}
```

VNS3 Controllers

GET /api/vns3_controllers

Get general list of VNS3 Controllers

Parameters:

- vns3_topology_id (optional) : ID of a particular topology
- vns3_controller_id (optional) : ID of a particular controller

Example:

```
$ curl -skL -H "api-token:foo" http://{server}/api/vns3_controllers
{
   "response" : [
      {
         "id" : 1,
         "name" : "EC2 test",
         "owner" : "admin",
         "virtual_network_id" : 82,
         "virtual_network" : "General Test",
         "vns3_topology_id" : 68,
         "vns3_topology" : "General Topology",
         "vns3_address" : "34.198.154.174",
         "private_ip_address" : "10.20.30.58",
         "vns3_version" : "3.5.2-20160914",
         "active" : true,
         "licensed" : true,
         "peered" : true,
         "description" : "",
         "controller_status" : {
            "last_contact_time" : "2017-09-25T16:27:46.000Z",
            "last_successful_contact_time" : "2017-09-25T16:27:46.000Z",
            "last_contact_code" : 200,
            "failed_contact_count" : 0,
            "last_contact_result" : "OK"
         },
      }
      {
         ...
      }
   ]
}
```

Notes: Using the vns3_controller_id is supported but we recommend use of the /api/vns3_controllers/:id GET method. The code path hit is smaller.

GET /api/vns3_controllers/:id

Get details about a specific VNS3 Controller based on ID

Parameters:

- id (required) : Controller ID from VNS3:ms

Example:

```
$ curl -skL -H "api-token:foo" http://{server}/api/vns3_controllers/1
{
   "response" : {
      "id" : 1,
      "name" : "EC2 test",
      "owner" : "admin",
      "virtual_network_id" : 82,
      "virtual_network" : "General Test",
      "vns3_topology_id" : 68,
      "vns3_topology" : "General Topology",
      "vns3_address" : "34.198.154.174",
      "private_ip_address" : "10.20.30.58",
      "vns3_version" : "3.5.2-20160914",
      "active" : true,
      "licensed" : true,
      "peered" : true,
      "description" : "",
      "controller_status" : {
         "last_contact_time" : "2017-09-25T16:27:46.000Z",
         "last_successful_contact_time" : "2017-09-25T16:27:46.000Z",
         "last_contact_code" : 200,
         "failed_contact_count" : 0,
         "last_contact_result" : "OK"
      },
   }
}
```

POST /api/vns3_controllers

Create a new VNS3 controller

Parameters:

- name (required) : Controller name

•vns3_topology_id (required) : Parent VNS3 topology ID

•address (required) : IP address of Controller

•api_v1_key (optional) : Secret key (password) for Controller's APIv1

•api_v2_key (optional) : Secret key (password) for Controller's APIv1.5/APIv2

•api_v2_username (optional) : Username for Controller's APIv1.5/APIv2

•active (optional) : true/false (default: true)

•description (optional) : Controller description

Notes: At least one of api_v1_key and api_v2_key is required. If api_v2_key is set then api_v2_username is also required.

Example:

```
$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" -X POST -d '{"name":"VC test
#1","vns3_topology_id":"1","address":"10.20.30.40","api_v1_key":"bad_password","description":"Test
controller"}' http://{server}/api/vns3_controllers
{"response_type":"success","response":{"msg":"VNS3 Controller created","id":1}}
```

PUT /api/vns3_controllers/:id

Update specified VNS3 controller

Parameters:

•id (required) : Controller ID from VNS3:ms

•name (optional) : Controller name

•vns3_topology_id (optional) : Parent VNS3 topology ID

•address (optional) : IP address of Controller

•api_v1_key (optional) : Secret key (password) for Controller's APIv1

•api_v2_key (optional) : Secret key (password) for Controller's APIv1.5/APIv2

•api_v2_username (optional) : Username for Controller's APIv1.5/APIv2

•active (optional) : true/false

•description (optional) : Controller description

Notes: If api_v2_key is set then api_v2_username is also required.

Example:

```
$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" -d '{"name":"foo",
"description":"bar"}' -X PUT http://{server}/api/vns3_controllers/1
{"response_type":"success","response":"Updated VNS3 controller"}
```

DELETE /api/vns3_controllers/:id

Delete specified VNS3 Controller

Parameter:

- id (required) : VNS3 Controller ID

Example:

```
$ curl -H "api-token:foo" -X DELETE http://localhost:43000/api/vns3_controllers/1
{"response_type":"success","response":"Deleted VNS3 Controller 1"}

$ curl -H "api-token:foo" -X DELETE http://localhost:43000/api/vns3_controllers/1
{"response_type":"error","response":"VNS3 Controller with ID 1 does not exist"}
```

GET /api/vns3_controllers/:id/status

Retrieve the status and live details of the selected VNS3 controller

Parameters:

- id (required) : Controller ID from VNS3:ms

Example:

```
$ curl -skL -H "api-token:foo" http://{server}/api/vns3_controllers/1/status
{"response_type":"success","response":{"manager_accessible":true,"controller_status":
{"last_contact_time":"2015-09-10T20:36:13.207Z","last_contact_result":"OK"},"system_config":
{"private_ipaddress":"10.20.30.40","public_ipaddress":"","topology_checksum":"c70c8f2...","vns3_versio
n":"3.5.0.9","topology_name":"test1","licensed":true,"peered":true,"asn":65001,"manager_id":1,"overlay
_ipaddress":"172.64.1.250"},"system_status":{"data":{"2015-09-09":{},"2015-09-10":
{}},"timestamp":"2015-09-10 21:36:13 +0100","timestamp_i":1441917373,"uptime":12075,"loadavg":
["0.06","0.07","0.06","1/187","31792"],"diskinfo":
[["/dev/sda1","10564440064","6073417728","3954393088","61%","/"]],"meminfo":
["514244608","495669248","18575360","0","41996288","217473024"],"swapinfo":
["1073737728","0","1073737728"],"container_system":
{"container_system_running":true,"images_limit":5,"images_stored":0,"containers_limit":5,"containers_a
ctive":0,"container_network":"172.0.10.0/24"}},"connection_status":{"connected_clients":
{},"connected_subnets":[],"ipsec":{"1":
{"local_subnet":"172.64.0.0/22","remote_subnet":"50.40.30.20/32","endpointid":1,"endpoint_name":"Foo",
```

```
"active":true,"description":"Tunnel foo","connected":false,"tunnel_params":{}}}},"peering_links":
{"peered":true,"managers":{"1":{"self":true,"id":1},"2":{"not_set":true,"id":2},"3":
{"not_set":true,"id":3},"4":{"not_set":true,"id":4}},"id":1}}}
```

PUT /api/vns3_controllers/:id/update_api_password

Update API password for specified VNS3 controller

Parameters:

- id (required) : Controller ID from VNS3:ms
- api_password (required) : Controller API password (will update controller)

Example:

```
$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" -d '{"api_password":"foo"}' -X PUT
http://{server}/api/vns3_controllers/1/update_api_password
{"response_type":"success","response":"API password updated"}
```

Notes: Updates both the VNS3:ms and the VNS3 controller instance

---

PUT /api/vns3_controllers/:id/update_ui

Update UI details for specified VNS3 controller

Parameters:

- id (required) : Controller ID from VNS3:ms
- username (optional) : Controller User
- password (optional) : Controller API password (will update controller)
- ui_enabled (optional) : Enable/disable UI (values: true/false)

Example:

```
$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" -d '{"password":"foo",
"ui_enabled":false}' -X PUT http://{server}/api/vns3_controllers/1/update_ui
{"response_type":"success","response":"UI details updated"}
```

Notes: Updates both the VNS3:ms and the VNS3 controller instance

---

GET /api/vns3_controllers/:id/ha

Get HA details for specified VNS3 controller

Parameters:

- id (required) : Controller ID from VNS3:ms
- remote_check (optional) : Check the remote servers for their status values rather than trust on the DB (can block). Default: false

Example:

```
$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" http://
{server}/api/vns3_controllers/1/ha
{"response_type":"success","response":
{"ha_enabled":true,"ha_backup_ip_address":"172.16.174.180","ha_uuid":"5e661f1dd656e1a0f3f3d9de3618da9c
806a665347a9e60684bdf519be23cf05","primary_sync_state":"queued","primary_sync_updated_at":"2015-05-07
10:23:00 +0100",ha_cloud_cred_id": 1}}

$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" http://
{server}/api/vns3_controllers/3/ha?remote_check=true
{"response_type":"success","response":
{"ha_enabled":false,"ha_initialised":false,"ha_cloud_validated":false,"ha_backup_ip_address":"172.16.1
74.182","ha_uuid":"806d9de3618d69be234bdf515e661dd656e10a96a6a0cf05568f3f3f1347a9ec","ha_cloud_cred_id
": 14}}
```

Notes: Additional information may be returned. Null values are removed. The full list of possible return variables is:

- ha_enabled
- ha_uuid
- ha_backup_ip_address
- primary_sync_state
- primary_sync_code
- ms_sync_state
- ms_sync_fail_count
- backup_active
- backup_sync_state
- backup_sync_code
- backup_sync_fail_count
- ha_cloud_cred_id

- primary_sync_updated_at
- ms_sync_updated_at
- backup_sync_updated_at
- activation_state
- stop_old_primary

PUT /api/vns3_controllers/:id/ha

Update HA details for specified VNS3 controller

Parameters:

- id (required) : Controller ID from VNS3:ms
- ha_enabled (required) : true/false
- If ha_enabled is true:
    - ha_type (required) : One of three strings: "hot", "warm", "cold"
    - ha_cloud_cred_id (required) : Cloud Credential ID
    - stop_old_primary (optional) : true/false - defaults to false
- If ha_type is "hot":
    - ha_uuid (required) : UUID value of backup instance
    - ha_backup_ip_address (required) : IP address of HA backup server
    - ha_api_v1_key (optional) : API v1 key (if different from Primary's)
    - ha_api_v2_username (optional) : API v2 (v1.5) username
    - ha_api_v2_key (optional) : API v2 (v1.5) key
- If ha_type is "warm":
    - ha_instance_id (required) : Instance ID of "warm" backup instance
- If ha_type is "cold":
    - ha_image_id (required) : Image ID for "cold" backup instance

Note: If ha_type is "hot" then one of ha_api_v1_key and ha_api_v2_key may be required. If ha_api_v2_key is used then ha_api_v2_username is required. The ha_api_v2_username and ha_api_v2_key fields are for use with VNS3 versions with APIv1.5 and newer enabled.

Example:

```
$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" -d
'{"ha_enabled":true,"ha_type":"hot","ha_uuid":"5e661f1dd656e1a0f3f3d9de3618da9c806a665347a9e60684bdf51
9be23cf05","ha_backup_ip_address":"10.20.30.40","ha_cloud_cred_id":14}' -X PUT http://
{server}/api/vns3_controllers/1/ha
{"response_type":"success","response":{"msg":"HA details updated","ha_enabled":true}}

$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" -d '{"ha_enabled":false}' -X PUT
http://{server}/api/vns3_controllers/1/ha
{"response_type":"success","response":{"msg":"HA details updated","ha_enabled":false}}
```

PUT /api/vns3_controllers/:id/ha/validate

Validate HA details for specified VNS3 controller. Make certain servers are accessible and the cloud details are compatible.

Parameters:

- id (required) : Controller ID from VNS3:ms

Example:

```
$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" -X PUT http://
{server}/api/vns3_controllers/1/ha/validate
{"response_type":"success","response":{"msg":"HA details fully
validated","ha_enabled":true,"ha_validated":true}}
```

If HA is not enabled then a success message will be returned with a warning as the message. The previous value of validation will be returned.

```
$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" -X PUT http://
{server}/api/vns3_controllers/1/ha/validate
{"response_type":"success","response":{"msg":"HA not enabled; details not
validated","ha_enabled":false,"ha_cloud_validated":true}}
```

Notes: This functionality has been separated from save as the action can be blocking. The process requires validation of the cloud details which requires both primary and backup servers be accessible.

PUT /api/vns3_controllers/:id/ha/initialise

Initialise HA for specified VNS3 controller and its backup device, set up pairing and configuration

Parameters:

•id (required) : Controller ID from VNS3:ms

Example:

```
$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" -X PUT http://
{server}/api/vns3_controllers/1/ha/initialise
{"response_type":"success","response":"HA configuration initialised"}}
```

PUT /api/vns3_controllers/:id/ha/sync

Trigger HA sync (via the queueing mechanism)

Parameters:

•id (required) : Controller ID from VNS3:ms

•sync_type (required) : String defining type of sync: 'full', 'pull' or 'push'

Example:

```
$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" -X PUT -d '{"sync_type":"full"}'
http://{server}/api/vns3_controllers/1/ha/sync
{"response_type":"success","response":"HA sync request queued"}}
```

Notes: The type of sync causes a change in workflow logic. A 'pull' triggers the creation of a syncfile on the primary server and pulls the resulting file to the local system; a 'push' sends the latest file from the management system to the backup server; a 'full' attempts to do both 'pull' and 'push' in sequence.

PUT /api/vns3_controllers/:id/ha/activate

Trigger the HA switchover activation

Parameters:

•id (required) : Controller ID from VNS3:ms

Example:

```
$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" -X PUT http://
{server}/api/vns3_controllers/1/ha/activate
{"response_type":"success","response":{status:"Running",message:"HA activation request
queued","timestamp":"2015-10-23 19:02:26 UTC","log_msgs":[{"timestamp":"2015-10-
23T19:02:26.000Z","status_message":"Activating","status":"success"}]}}
```

GET /api/vns3_controllers/:id/ha/activate

Get current status of HA switchover activation

Parameters:

- id (required) : Controller ID from VNS3:ms

Example:

```
$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" -X GET http://
{server}/api/vns3_controllers/1/ha/activate
{"response_type":"success","response":{"status":"Activating","timestamp":"2015-10-23 20:43:50
UTC","log_msgs":[{"timestamp":"2015-10-
23T20:41:15.000Z","status_message":"Activating","status":"success"},{"timestamp":"2015-10-
23T20:41:19.000Z","status_message":"Static IP is being remapped","status":"success"},
{"timestamp":"2015-10-23T20:41:21.000Z","status_message":"Updating Cloud Route
Tables","status":"success"}]}}
```

Notes: status values in the log are "success" or "fail".

Cloud VLANs

GET /api/cloud_vlans

Get general list of Cloud VLANs

Parameters:

- cloud_vlan_id (optional) : ID of a particular cloud vlan

Example:

```
$ curl -H "api-token:foo" http://{server}/api/cloud_vlans
```

Notes: Using the cloud_vlan_id is supported but we recommend use of the /api/cloud_vlans_/:id GET method. The database hit is smaller.

GET /api/cloud_vlans/:id

Get details about a specific Cloud VLAN based on ID

Example:

```
$ curl -H "api-token:foo" http://{server}/api/cloud_vlans/1
```

POST /api/cloud_vlans

Create a new Cloud VLAN

Parameters:

- name (required) : Cloud VLAN name
- virtual_network_id (required) : Parent virtual network ID
- description (optional) : Cloud VLAN description

Example:

```
$ curl -skL -H "api-token:foo" -H "Content-Type:application/json" -X POST  -d '{"name":"VLAN test #1",
"virtual_network_id":"1", "description":"Test VLAN"}' http://{server}/api/cloud_vlans
{"response_type":"success","response":{"msg":"Cloud VLAN created","id":13}}
```

PUT /api/cloud_vlans/:id

Update specified Cloud VLAN

Parameters: (All except ID are optional)

- id (required) : Cloud VLAN ID from VNS3:ms
- name : Cloud VLAN name
- virtual_network_id : Parent virtual network ID
- description : Cloud VLAN description

Example:

```
$ curl -s -H "api-token:foo" -H "Content-Type:application/json" -d '{"name":"foo",
"description":"bar"}' -X PUT http://{server}/api/cloud_vlans/1
{"response_type":"success","response":"Updated Cloud VLAN"}
```

DELETE /api/cloud_vlans/:id

Delete specified Cloud VLAN

Parameter:

- id (required) : Cloud VLAN ID

Example:

```
$ curl -H "api-token:foo" -X DELETE http://{server}/api/cloud_vlans/1
{"response_type":"success","response":"Deleted Cloud VLAN 1"}

$ curl -H "api-token:foo" -X DELETE http://{server}/api/cloud_vlans/1
{"response_type":"error","response":"Cloud VLAN with ID 1 does not exist"}
```

Cloud VLAN Components

GET /api/cloud_vlan_components

Get general list of Cloud VLAN Components

Parameters:

- cloud_vlan_component_id (optional) : ID of a particular Cloud VLAN component

Example:

```
$ curl -H "api-token:foo" http://{server}/api/cloud_vlan_components
{"response":[{"id":1,"name":"172.16.174.180","vns3_topology":"VNS3 Topology
1","vns3_address":"172.16.174.180","private_ip_address":"172.16.174.180","vns3_version":"3.5.0.00-
20140430","licensed":true,"active":true,"peered":true,"description":"vns3test-180 edit"},
{"id":2,"name":"172.16.174.181","vns3_topology":"VNS3 Topology 1","vns3_address":"vns3test-
181","private_ip_address":"172.16.174.181","vns3_version":"3.5.0.00-
20140430","licensed":true,"active":true,"peered":true,"description":"bar"}]}
```

Notes: Using the cloud_vlan_component_id is supported but we recommend use of the /api/cloud_vlan_components/:id GET method. The database hit is smaller.

GET /api/cloud_vlan_components/:id

Get details about a specific Cloud VLAN Component based on ID

Example:

```
$ curl -H "api-token:foo" http://{server}/api/cloud_vlan_components/1
{"response":{"id":1,"name":"172.16.174.180","vns3_topology":"VNS3 Topology
1","vns3_address":"172.16.174.180","private_ip_address":"172.16.174.180","vns3_version":"3.5.0.00-
20140430","licensed":true,"active":true,"peered":true,"description":"vns3test-180 edit"}}
```

POST /api/cloud_vlan_components

Create a new Cloud VLAN Component

Parameters:

- name (required) : Cloud VLAN name
- cloud_vlan_id (required) : Parent Cloud VLAN ID
- user_cred_id (required) : User's cloud credential ID

•vlan_component_id (required) : Cloud's ID of component to be tracked

•region (optional) : Region for component to be tracked (if applicable)

•description (optional) : Cloud VLAN Component description

Example:

```
$ curl -H "api-token:foo" -H "Content-Type:application/json" -X POST  -d '{"name":"component test #1",
"cloud_vlan_id":"14", "user_cred_id":"1", "vlan_component_id":"vpc-01234567", "description":"Test VLAN
component"}'  http://localhost:43000/api/cloud_vlan_components
{"response_type":"success","response":{"msg":"Cloud VLAN Component created","id":1}}
```

PUT /api/cloud_vlan_components/:id

Update specified Cloud VLAN Component

Parameters: (All except ID are optional)

•id (required) : Cloud VLAN Component ID from VNS3:ms

•name : Cloud VLAN Component name

•cloud_vlan_id : Parent Cloud VLAN ID

•user_cred_id : User cloud credential ID

•region : cloud region string

•description : Cloud VLAN Component description

Example:

```
$ curl -s -H "api-token:foo" -H "Content-Type:application/json" -d '{"name":"foo",
"description":"bar"}' -X PUT http://{server}/api/cloud_vlan_components/1
{"response_type":"success","response":"Updated VNS3 Cloud VLAN component"}
```

DELETE /api/cloud_vlan_components/:id

Delete specified Cloud VLAN Component

Parameter:

•id (required) : Cloud VLAN Component ID

Example:

```
$ curl -H "api-token:foo" -X DELETE http://{server}/api/cloud_vlan_components/1
```

```
{"response_type":"success","response":"Deleted Cloud VLAN Component 1"}
$ curl -H "api-token:foo" -X DELETE http://{server}/api/cloud_vlan_components/1
{"response_type":"error","response":"Cloud VLAN Component with ID 1 does not exist"}
```

Backups

GET /api/backups

Return a list of available database backup files

Example:

```
$ curl -H "api-token:foo" http://{server}/api/backups
{"response_type":"success","response":{"backup_files":[{"backup_name":"backup-20141014-
093455.gz","create_time":"2014-10-14T09:34:55.732-05:00","size":"10000000"},{"backup_name":"backup-
20141106-101613.gz","create_time":"2014-11-06T10:16:13.555-
06:00","size":"10000000"}],"failed_backups":[{"backup_name":"backup-20150106-
130203.failed","create_time":"2015-01-06T13:02:03.604-06:00","size":"0"},{"backup_name":"backup-
20150106-130115.failed","create_time":"2015-01-06T13:01:15.724-06:00","size":"0"}]}}
```

GET /api/backups/download

Download a named backup file

Parameters:

- backup_name (required) : Backup filename

Example:

```
$ curl -H "api-token:foo" http://{server}/api/backups/download?backup_name=backup.gz >
output_backup.gz
```

Notes: Return value is the GZIPed backup file.

POST /api/backups/upload

Upload a named backup file as a form element

Parameters:

- backup_file (required) : Backup file

Examples:

```
$ curl -X POST -H "api-token:foo" --form backup_file=@foo.gz http://{server}/api/backups/upload
{"response_type":"success","response":"Backup file 'foo.gz' uploaded"}
$ curl -X POST -H "api-token:foo" --form backup_file=@foo.gz http://{server}/api/backups/upload
{"response_type":"error","response":"File exists"}
```

```
$ curl -X POST -H "api-token:foo" --form backup_file=@foo http://{server}/api/backups/upload
{"response_type":"error","response":"Invalid file extension"}
```

DELETE /api/backups

Delete a backup file

Parameters:

- backup_name (required) : Backup filename

Example:

```
$ curl -X DELETE -H "api-token:foo" -H "Content-Type:application/json" -d
'{"backup_name":"XEMPTY.gz"}' http://{server}/api/backups
{"response_type":"success","response":"Backup file 'backup.gz' deleted"}
```

POST /api/backups/create_backup

Schedule a backup creation

Example:

```
$ curl -H "api-token:foo" -X POST http://{server}/api/backups/create_backup
{"response_type":"success","response":{"status":"Backup process queued","uuid":"29643b97-7502-40fd-
bbe1-4d955e1816dc"}}
```

Note: Retrieve the backup job status from GET /api/system/jobs

---

POST /api/backups/restore_backup

Restore a backup file, resetting state of VNS3:ms

Parameters:

- backup_name (required) : Backup filename

Example:

```
$ curl -X POST -H "api-token:foo" -H "Content-Type:application/json" -d '{"backup_name":"backup.gz"}'
http://{server}/api/backups/restore_backup
{"response_type":"success","response":"Backup restored"}
```

Notes: File must already exist locally (on the VNS3:ms box). If necessary use upload API function to upload first.

---

snapshots_backup

GET /api/snapshots_backup

Return details of snapshots backup

Example:

```
$ curl -H "api-token:foo" -X GET http://{server}/api/snapshots_backup
{"response_type":"success","response":{"snapshot_backup_file":{"backup_name":"backup-20150710-
143452.snapshots","create_time":"2015-07-10T14:53:40.615-05:00","size":337715744}}}
```

GET /api/snapshots_backup/download

Download the snapshot backup file

Example:

```
$ curl -H "api-token:foo" http://{server}/api/snapshots_backup/download > output_backup
```

Notes: Return value is the encrypted backup file.

DELETE /api/snapshots_backup

Delete the snapshots backup file

Example:

```
$ curl -X DELETE -H "api-token:foo" http://{server}/api/snapshots_backup
{"response_type":"success","response":"Backup file backup-20150710-143452.snapshots deleted"}
$ curl -X DELETE -H "api-token:foo" http://{server}/api/snapshots_backup
{"response_type":"error","response":"No backup file"}
```

POST /api/snapshots_backup/create_backup

Schedule a backup creation

Example:

```
$ curl -H "api-token:foo" -X POST http://{server}/api/snapshots_backup/create_backup
```

```
{"response_type":"success","response":{"status":"Backup process queued","uuid":"ca083baf-5fc1-43f4-9d4e-aa9f1ccd4fff"}}
```

Note: Retrieve the snapshots backup job status from GET /api/system/jobs

---

POST /api/snapshots_backup/upload_backup

Schedule a 'pull' to retrieve a snapshots backup file from a specified URL

Parameters:

- url (required) : URL to use

Example:

```
$ curl -H "Content-Type: application/json" -H "api-token:foo" -X POST -d '{"url":"http://server/path/file.snapshots"}' http://{server}/api/snapshots_backup/upload_backup
{"response_type":"success","response":{"status":"Snapshots file retrieval queued","uuid":"e51d1b15-6195-4370-905c-12706ca8204b"}}
```

Notes: Upon success any previous backup file will be replaced in line with the standard of only locally storing one snapshots backup file Retrieve the snapshots backup job status from GET /api/system/jobs

---

POST /api/snapshots_backup/restore_backup

Restore a backup file, restoring/replacing snapshot files

Parameters:

- backup_name (required) : Backup filename

Example:

```
$ curl -X POST -H "api-token:foo" -H "Content-Type:application/json" -d '{"backup_name":"backup.snapshots"}' http://{server}/api/snapshots_backup/restore_backup
{"response_type":"success","response":"Snapshot backup restored"}
```

Notes: The file must already exist on the VNS3:ms server. If necessary use upload_backup API function to pull a snapshots backup file to the VNS3:ms server first.

Snapshots

GET /api/snapshots

Return a list of available snapshot files grouped by VNS3 controller (optionally filtered by VNS3 controller)

Parameters:

- vns3_controller_id (optional) : VNS3 controller ID
- vns3_topology_id (optional) : VNS3 topology ID
- snapshot_id (optional) : Specific snapshot ID

Example:

```
$ curl -H "api-token:foo" http://{server}/api/snapshots?vns3_controller_id=1
{"response":{"snapshots":[{"snapshot_id":75,"created_at":"2014-10-
29T20:00:35.000Z","snapshot_name":"snapshot_20141029_1414612836_38.88.227.198","snapshot_size":3088854
,"sha1_checksum":"0d3197c5d4f5ebaff50149ee81ec3a9a9226da46","available":true,"status":"Ready","vns3_co
ntroller_id":1,"vns3_controller_name":"172.16.174.180"}],"failed_snapshots":
[{"snapshot_id":47,"created_at":"2014-10-27T12:24:18.000Z","snapshot_name":"Failed
snapshot","snapshot_size":null,"sha1_checksum":null,"available":false,"status":"Snapshot creation
failed","vns3_controller_id":1,"vns3_controller_name":"172.16.174.180"}]}}
```

Notes: Use at most only one of snapshot_id, vns3_controller_id, vns3_topology_id. Only snapshot records that have available (non-deleted) VNS3 controllers will be returned.

GET /api/snapshots/download

Download a specific snapshot file

Parameters:

- snapshot_id (required) : Snapshot ID

Example:

```
$ curl -H "api-token:foo" http://{server}/api/snapshots/download?snapshot_id=1234 > output_snapshot.gz
```

Notes: Return value is the encrypted snapshot archive.

POST /api/snapshots

Create a snapshot for a particular VNS3 Controller

Parameters:

- vns3_controller_id (required) : VNS3 Controller ID

Example:

```
$ curl -H "api-token:foo" -X POST http://{server}/api/snapshots?vns3_controller_id=1
{"response_type":"success","response":"Snapshot created"}

$ curl -H "api-token:foo" -X POST http://{server}/api/snapshots?vns3_controller_id=1
{"response_type":"error","response":"Maximum configuration snapshots reached."}

$ curl -H "api-token:foo" -X POST http://{server}/api/snapshots?vns3_controller_id=2
{"response_type":"error","response":"Unable to connect to VNS3 Controller"}

$ curl -H "api-token:foo" -X POST http://{server}/api/snapshots?vns3_controller_id=299
{"response_type":"error","response":"Missing VNS3 Controller with ID 299"}
```

DELETE /api/snapshots

Delete one or more snapshot files

Parameters:

- snapshot_ids (required) : Array of snapshot IDs

Example:

```
$ curl -H "api-token:foo" -X DELETE -d '{"snapshot_ids": [1,2,3]}' http://{server}/api/snapshots
{"response_type":"success","response":"Accessible snapshots deleted"}
```

User

GET /api/user/display_options

Return current user display options list

Example:

```
$ curl -kL -H "api-token:foo" -X GET http://{server}/api/user/display_options
{"tree_alphabetise":false,"tree_insert_at_top":false}
```

PUT /api/user/display_options

Update current user display option values

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type:application/json" -d '{"tree_alphabetise":true}' -X PUT
http://{server}/api/user/display_options
{"response_type":"success","response":"Updated user display options"}

$ curl -kL -H "api-token:foo" -H "Content-Type:application/json" -d
'{"tree_alphabetise":true,"tree_insert_at_top":true}' -X PUT http://{server}/api/user/display_options
{"response_type":"error","response":"Alphabetical sorting and top-of-tree insertion are mutually
exclusive. At most only one can be enabled."}
```

Notes: Payload can be any combination of options, must send at least one. Full set of options can be determined via the GET /api/user/display_options call.

PUT /api/user/regenerate_api_token

Regenerate and return API token

Parameters:

- new_token (optional) : A user-defined new token

Example:

```
$ curl -kL -H "api-token:foo" -X PUT -H "Content-Type:application/json" -d '{"new_token":"bar"}'
http://{server}/api/user/regenerate_api_token
{"response_type":"success","response":{"api-token":"bar"}}

$ curl -kL -H "api-token:bar" -X PUT http://{server}/api/user/regenerate_api_token
```

```
{"response_type":"success","response":{"api-
token":"8c2f7106149b5eaea5277ab26aa988c556c6c71a40b53ccef6a9635583f666ec"}}
```

Notes:

- Only the superuser can regenerate the API token with a user-defined value
- If a token is user-defined (via the parameter) then it will be flagged as a "default" token and will limit access to API functionality.

PUT /api/user/password

Update the user password

Parameters:

- password (required) : New password
- force_refresh (optional) : Require user update password on next login

Example:

```
$ curl -kL -H "api-token:foo" -X PUT -H "Content-Type:application/json" -d '{"password":"newpw"}'
http://{server}/api/user/password
{"response_type":"success","response":"Password updated"}
```

Notes: If the user is logged in to the UI they will be logged out.

GET /api/user/validate_username

Check a username ensuring that it is not already in use

Parameters:

- value (required): username to validate

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type:application/json" -d '{"value":"new_user"}' -X GET
http://{server}/api/user/validate_email
{"response_type":"success","response":"Username valid"}

$ curl -kL -H "api-token:foo" -H "Content-Type:application/json" -d '{"value":"admin"}' -X GET http://
{server}/api/user/validate_email
{"response_type":"error","response":"Username already in use"}
```

Notes: A username will always be tagged as a duplicate if it already exists

---

GET /api/user/validate_email

Check an e-mail address ensuring that it is valid and not already in use (other than with the current user)

Parameters:

- value (required): e-mail address to validate

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type:application/json" -d '{"value":"foo@bar.com"}' -X GET
http://{server}/api/user/validate_email
{"response_type":"success","response":"E-mail valid"}

$ curl -kL -H "api-token:foo" -H "Content-Type:application/json" -d '{"value":"foo2@bar.com"}' -X GET
http://{server}/api/user/validate_email
{"response_type":"error","response":"E-mail already in use"}

$ curl -kL -H "api-token:foo" -H "Content-Type:application/json" -d '{"value":"foo.bar"}' -X GET
http://{server}/api/user/validate_email
{"response_type":"error","response":"Invalid e-mail format"}
```

Notes: An e-mail value will not be defined as a duplicate if it belongs to the calling user.

GET /api/user/credentials

Return list of user credentials

Parameters:

- cred_id (optional) : integer value of credential ID
- fields (optional) : boolean value, default: false

Example:

```
$ curl -kL -H "api-token:foo" -X GET http://{server}/api/user/credentials?cred_id=1&fields=false
{"response_type":"success","response":[{"id":"1","name":"1234-5678-
9012","code":"ec2","verified":true,"verification_message":"Cloud creds verified"}]}

$ curl -kL -H "api-token:foo" -X GET http://{server}/api/user/credentials?cred_id=1&fields=true
{"response_type":"success","response":[{"id":"1","name":"1234-5678-
9012","code":"ec2","verified":true,"verification_message":"Cloud creds verified","fields":
[{"key":"Account ID","value":"1234-5678-9012"},{"key":"Access Key","value":"XXXXXXXXXXXXXXXXXXX"},
{"key":"Secret Key","value":"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"},
{"key":"GovCloud","value":"0"}]}]}
```

POST /api/user/credentials

Create a new user credential

Parameters:

- •name (required) : Name of credentials
- •code (required) : Credential type code
- •fields (required) : An array of hashes containing key/value pairs for the credential fields.

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X POST -d '{"name":"1234-5678-
9012","code":"ec2","fields":[{"key":"Access Key","value":"XXXXXXXXXXXXXXXXXXX"},{"key":"Secret
Key","value":"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"},{"key":"GovCloud","value":'true'}]}' http://
{server}/api/user/credentials
{"response_type":"success","response":"User credentials saved. Validating.","id":1}
```

Notes: See /api/system/credential_types/:code for the list of fields to be sent based on the code. The key is the field name and the value is to be defined by the caller. After the credentials have been saved they will be validated. To examine the results use the GET /api/user/credentials?cred_id=ID call.

DELETE /api/user/credentials/:id

Delete user credentials

Parameters:

- •id (required)

Example:

```
$ curl -kL -H "api-token:foo" -X DELETE http://{server}/api/user/credentials/99
{"response_type":"success","response":"Credentials deleted"}
```

PUT /api/user/credentials/:id

Update user credentials

Parameters:

- •id (required)
- •name (optional) : Name of credentials
- •code (required) : Credential type code

•fields (required) : An array of hashes containing key/value pairs for the credential fields.

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X PUT -d '{"name":"1234-5678-
9012","code":"ec2","fields":[{"key":"Access Key","value":"XXXXXXXXXXXXXXXXXXX"},{"key":"Secret
Key","value":"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"},{"key":"GovCloud","value":'true'}]}' http://
{server}/api/user/credentials/1
{"response_type":"success","response":"User credentials updated. Validating.","id":1}
```

Notes: If the code is changed then the entire set of associated credential type key/value pairs must be sent. If the code is remaining the same then only changed values need to be sent.

Admin

PUT /api/admin/ldap

Enable/disable LDAP

Parameters:

- enabled (required) : boolean true or false

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X PUT -d '{"enabled":true}' http://
{server}/api/admin/ldap
{"response_type":"success","response":{"ldap_enabled":true}}
```

GET /api/admin/ldap/settings

Retrieve LDAP connection settings

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X GET http://
{server}/api/admin/ldap/settings
{"response_type":"success","response":
{"ldap_host":"10.20.30.40","ldap_port":"389","ldap_ssl":false,"ldap_binddn":"dc=cfttest,dc=com","ldap_
bindpw":""}}
```

PUT /api/admin/ldap/settings

Set LDAP connection settings

Parameters:

- ldap_host (required) : IP address or resolvable hostname
- ldap_port (optional) : Port to use - default: 389
- ldap_ssl (optional) : Use SSL? True/False.
- ldap_binddn (optional) : Bind Username
- ldap_bindpw (optional) : Bind Password

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X PUT -d
'{"ldap_host":"10.20.30.40","ldap_port":"389","ldap_ssl":false,"ldap_binddn":"dc=cfttest,dc=com","ldap
_bindpw":""}' http://{server}/api/admin/ldap/settings
{"response_type":"success","response":"Settings updated"}
```

POST /api/admin/ldap/settings

Validate the LDAP connection

Parameters:

- ldap_host (required) : LDAP hostname
- ldap_port (optional) : LDAP server port (default: 389)
- ldap_ssl (required) : Use SSL, true/false
- ldap_binddn (optional) : Username
- ldap_bindpw (optional) : Password

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X POST -d
'{"ldap_host":"10.20.30.40", "ldap_ssl":false,"ldap_binddn","dc=test,dc=com"}' http://
{server}/api/admin/ldap/test_connection
{"response_type":"success","response":{"connect_success":true}}
```

GET /api/admin/ldap/user_schema

Retrieve LDAP user schema settings

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X GET http://
{server}/api/admin/ldap/user_schema
{"response_type":"success","response":
{"ldap_user_base":"dc=cfttest,dc=com","ldap_user_id_attribute":"uid","ldap_user_list_filter":null}}
```

PUT /api/admin/ldap/user_schema

Set LDAP user schema settings

Parameters:

•ldap_user_base (required) : Base DN from which to search for Users

•ldap_user_id_attribute (required) : Attribute type for the Users

•ldap_user_list_filter (optional) : Search filter for Users - defaults to '*'

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X PUT -d
'{"ldap_user_base":"dc=cfttest,dc=com","ldap_user_id_attribute":"uid"}' http://
{server}/api/admin/ldap/user_schema
{"response_type":"success","response":"LDAP user schema settings updated"}

$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X PUT -d
'{"ldap_user_base":"dc=cfttest,dc=com","ldap_user_id_attribute":"uid","ldap_user_list_filter":"x"}'
http://{server}/api/admin/ldap/user_schema
{"response_type":"error","response":"Invalid filter syntax error. Filter must follow format specified
in RFC-2254"}
```

Notes: The user list filter string must be in the prefix syntax specified in RFC-2254 else an error message will be returned

---

POST /api/admin/ldap/user_schema

Validate LDAP schema settings by submitting values and retrieving associated data

Parameters:

•ldap_user_base (required) : Base DN from which to search for Users

•ldap_user_id_attribute (required) : Attribute type for the Users

•ldap_user_list_filter (optional) : Search filter for Users - defaults to '*'

•limit (optional) : Maximum number of records to return (default: 100)

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X POST -d
'{"ldap_user_base":"dc=cfttest,dc=com","ldap_user_id_attribute":"uid","limit":2}' http://
{server}/api/admin/ldap/test_user_schema
{"response_type":"success","response":["test_user_1","test_user_2"]}
```

GET /api/admin/ldap/group_schema

Retrieve LDAP group schema settings

Example: If LDAP groups are enabled/required:

```
$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X GET http://
{server}/api/admin/ldap/group_schema
{"response_type":"success","response":
{"ldap_group_required":true,"ldap_group_base":"ou=vns3groups,dc=cfttest,dc=com","ldap_group_id_attribu
te":"cn","ldap_group_list_filter":"*","ldap_group_member_attribute":"member","ldap_group_member_attr_f
ormat":"UserDN","ldap_search_scope":"subtree"}}
```

If disabled:

```
$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X GET http://
{server}/api/admin/ldap/group_schema
{"response_type":"success","response":{"ldap_group_required":false}}
```

PUT /api/admin/ldap/group_schema

Set LDAP group schema settings

Parameters:

- ldap_group_required (required) : Are group searches required?

If ldap_group_required is true then the following apply:

- ldap_group_base (required) : Base DN from which to search for Groups
- ldap_group_id_attribute (required) : Attribute type for the Groups
- ldap_group_list_filter (optional) : Search filter for Groups - defaults to '*'
- ldap_group_member_attribute (required) : Attribute used to search for a user within the Group
- ldap_group_member_attr_format (optional) : Format of the Group Member attribute. Must be one of 'UserID' or 'UserDN'. Defaults to 'UserDN'
- ldap_search_scope (optional) : Depth of search. Must be one of 'base', 'single' or 'subtree'. Defaults to 'subtree'

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X PUT -d
'{"ldap_group_required":true,"ldap_group_base":"ou=vns3groups,dc=cfttest,dc=com","ldap_group_id_attrib
ute":"cn","ldap_group_list_filter":"*","ldap_group_member_attribute":"member","ldap_group_member_attr_
format":"UserDN"}' http://{server}/api/admin/ldap/group_schema
{"response_type":"success","response":"LDAP group schema settings updated"}
```

POST /api/admin/ldap/group_schema

Validate LDAP schema settings by submitting values and retrieving associated data

Parameters:

- ldap_group_base (required) : Base DN from which to search for Groups
- ldap_group_id_attribute (required) : Attribute type for the Groups
- ldap_group_list_filter (optional) : Search filter for Groups - defaults to '*'
- ldap_group_member_attribute (required) : Attribute used to search for a user within the Group
- ldap_group_member_attr_format (optional) : Format of the Group Member attribute. Must be one of 'UserID' or 'UserDN'. Default: 'UserDN'
- ldap_search_scope (optional) : Search scope. Must be one of 'base', 'single', or 'subtree'. Default: 'subtree'
- limit (optional) : Maximum number of records to return (default: 100)

Example:

```
$ curl -kL -H "api-token:foo" -H "Content-Type: application/json" -X POST -d
'{"ldap_group_base":"ou=vns3groups,dc=cfttest,dc=com","ldap_group_id_attribute":"cn","ldap_group_membe
r_attribute":"member","ldap_group_member_attr_format":"UserDN"}' http://
{server}/api/admin/ldap/group_schema
{"response_type":"success","response":[{"ldap_group":"vns3admin","ldap_users":["testadmin"]},
{"ldap_group":"vns3operator","ldap_users":["testoperator","testadmin"]},
{"ldap_group":"vns3user","ldap_users":["testoperator","testuser","testadmin"]}]}
```

Notes: Returns an array of groups and a sub-array of users for each group