VPN-Cubed Datacenter Connect

API Guide

v20101115

Requirements

You have an Amazon AWS account that CohesiveFT can use for enabling your access to the VPN-Cubed AMIs.

You have agreed to the terms of service provided for the VPN-Cubed AMIs.

Access to a deployment launch tool for launching the VPN-Cubed AMI

- ElasticFox
- AWS Console
- Rightscale

Ability to create/configure EC2 security groups using a tools like; ElasticFox, AWS Console, CohesiveFT's Elastic Server On-Demand, or EC2 Command Line tools

Basic knowledge of Linux software installation and use of command line tools.

This guide assumes you know how to configure and launch a VPN-Cubed Manager. If you do not please review those steps for your selected VPN-Cubed version and edition at www.cohesiveft.com/vpncubed

For support requests use our community forums at:
http://getsatisfaction.com/cohesiveft

or for other support inquiries including paid support services contact:
sales@cohesiveft.com

Your Configuration Begins Here!

- For you initial use of the API you will need access to a VPN-Cubed 2.0 AMI.
- You will also need to have available a snapshot of your existing manager configuration or you should make one for use in this API preview.
- The API uses a Ruby script and Ruby language binding for the API. You will need several Ruby libraries on your local machine where you will be making the API calls. Those are listed on the next page.

- On a Debian/Ubuntu Linux the commands are:
  - apt-get update
  - apt-get install ruby
  - apt-get install rubygems
  - apt-get install rake
  - apt-get install build-essential
  - apt-get install ruby1.8-dev
  - gem install json
  - apt-get install libopenssl-ruby

- Download the VPN-Cubed API tar/zip (includes “api.rb” and “vpncubed.rb”) available on all the VPN-Cubed pages under Launch Instructions.

VPN3 API: Calling Structure

VPN3 API calls take several arguments in common across all calls: “-K” for access key, “-S” for access secret, “-H” for the ip address of the manager you are connecting to.

Some of the calls can only be made after a manager is licensed or configured. This will be noted in the documentation for the API call.

Here is an example call that can be made at any time in a VPN3 Manager’s lifecycle.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address desc_config

vpn3api_demo: ruby $vpn3api_home/vpncubed.rb -K api -S i-6b344d01 -H 174.129.238.73 desc_config

Response:
public_ipaddress: 174.129.238.73
vpncubed_version: 0.7.9999.2-20100823171847
licensed: false
private_ipaddress: 10.220.23.235
topology_checksum: 3309dfc9832c45280b590341e39de1277c17ffbd

The manager is unlicensed at this point, meaning neither a license, nor a snapshot with a license has been provided to it. The private_ipaddress is the actual underlying LAN address provided by the cloud provider. The topology checksum was randomly generated, and changes with the license and/or snapshot info.

Basic Workflow Choices:

A) Use the web UI to “design” one or more networks. Take snapshots of the managers and start API testing from launching managers and calling “import_snapshot”

B) Start with launching manager(s). If using Free or Lite Editions start with “create_keyset”; if using SME or Enterprise Editions you will need to configure the instance(s) with a license provided by CFT and “upload_license”.

Notes:

- “Configurable Addressing” is not yet available via API. For testing configurations with configurable addressing use the Web UI to design addressing scheme and save to snapshot.

- There is some inconsistency in command line call arguments as strings; where it is sometimes unclear as an example to represent a CIDR as “192.168.1.0/24” or as 192.168.1.0/24, or if in fact both work.

API Command Line Calls

Command Line Call	Call Category	Valid Pre-License
reset_password	Admin	**Yes**
setup_remote_support	Admin	No
setup_admin_ui	Admin	No
server_action	Admin	No
desc_config	Status	**Yes**
desc_status	Status	No
desc_ipsec_status	Status	No
desc_client_status	Status	No
desc_system_status	Status	No
desc_link_history	Status	No
desc_license	License	No
upload_license	License	**Yes**
desc_keyset	Clientpacks	No
setup_keyset	Clientpacks	No
desc_clientpacks	Clientpacks	No
fetch_clientpacks	Clientpacks	No
get_next_available_clientpack	Clientpacks	No
edit_clientpack	Clientpacks	No
desc_peering	Peering	No
set_manager_id	Peering	No
add_peer	Peering	No
edit_peer	Peering	No
delete_peer	Peering	No
desc_snapshot	Snapshots	No
create_snapshot	Snapshots	No
delete_snapshot	Snapshots	No
import_snapshot	Snapshots	**Yes**
fetch_snapshot	Snapshots	No
desc_ipsec_status	IPsec	No
setup_ipsec	IPsec	No
get_ipsec_local_address	IPsec	No
set_ipsec_local_address	IPsec	No
create_ipsec_endpoint	IPsec	No
edit_ipsec_endpoint	IPsec	No
delete_ipsec_endpoint	IPsec	No
desc_ipsec_endpoint	IPsec	No
create_remote_subnet	IPsec	No
delete_remote_subnet	IPsec	No

Call: reset_password
Argument Switch: --password
Argument: Password as a string
Allowed Pre-License: Yes
Purpose:

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address reset_password --password “mysuperpassword”

ruby $vpn3api_home/vpncubed.rb -K api -S i-6b344d01 -H 174.129.238.73 reset_password --password "apidemopassword"

Response:
password_reset: ok

Note: Now that we have changed the password, we use this as our new “-S” argument.

Call: setup_remote_support
Argument Switch: --enabled (true or false)
Argument: The “--enabled” arg specifies whether remote support is enabled.
Allowed Pre-License: No
Purpose: Enables and disables remote support.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address setup_remote_support --enabled true

ruby $vpn3api_home/vpncubed.rb -K api -S "apidemopassword" -H 174.129.238.73 setup_remote_support --enabled true

Response:
TBD

Call: setup_admin_ui
Argument Switch: --enabled (true or false), --username (string), --password (string)
Argument: The “--enabled” arg specifies whether the web UI is enabled. The “--username” arg specifies the username for the web UI. The “--password” arg specifies the password for the web UI.
Allowed Pre-License: No
Purpose: Enables and disables the web UI. Allows the username and password for the web UI to be set.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address setup_admin_ui --enabled true --username newusername --password newpassword

ruby $vpn3api_home/vpncubed.rb -K api -S "apidemopassword" -H 174.129.238.73 setup_admin_ui --enabled true --username vpn3 -password vpn3

Response:
username: vpncubed
enabled: true

Call: server_action
Argument Switch: --reboot (true or false)
Argument: The “--reboot” arg specifies whether to reboot the VPN3 manager.
Allowed Pre-License: No
Purpose: Re-boots the manager.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address server_action --reboot true

ruby $vpn3api_home/vpncubed.rb -K api -S "apidemopassword" -H 174.129.238.73 server_action --reboot true

Call: desc_config
Argument Switch: None
Argument: None
Allowed Pre-License: Yes
Purpose: Provides general information about the manager's topology, license state and checksums

command prompt> ruby vpncubed.rb -K "api" -S "myapisecret" -H manager-ip-address desc_config

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.73 desc_ipsec_status

Response:

```
public_ipaddress: 174.129.238.73
vpncubed_version: 0.7.9999.2-20100823171847
licensed: false
private_ipaddress: 10.220.23.235
topology_checksum: 3309dfc9832c45280b590341e39de1277c17ffbd
```

Call: desc_status
Argument Switch: None
Argument: None
Allowed Pre-License: No
Purpose: Provides general information on all defined IPsec and client overlay connections

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address desc_status

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.73 desc_ipsec_status

Response: (list of connected clients, info about connected tunnels)

```
connected_clients:
  172.31.1.53:
    ipaddress: 10.160.139.146
    managerid: 3
    overlay_ipaddress: 172.31.1.53
 <more>
```

```
ipsec:
  192.168.1.0/24:
    tunnel_params:
      phase1_cipher: aes_256
      phase1: up
      nat_t: "on"
      phase2: up
      phase1_dh_group: 5
      dpd: "on"
      phase2_algo: AES_256-HMAC_SHA1
      phase1_prf: sha
    connected: true
    subnet: 192.168.1.0/24
    managerid: 2
    endpointid: 4
```

Call: desc_ipsec_status
Argument Switch: None
Argument: None
Allowed Pre-License: No
Purpose: Provides information on all defined IPsec connections

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address desc_ipsec_status

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.73 desc_ipsec_status

Response:

```
ipsec:
  192.168.1.0/24:
    tunnel_params:
      phase1_cipher: aes_256
      phase1: up
      nat_t: "on"
      phase2: up
      phase1_dh_group: 5
      dpd: "on"
      phase2_algo: AES_256-HMAC_SHA1
      phase1_prf: sha
    connected: true
    subnet: 192.168.1.0/24
    managerid: 2
    endpointid: 4
```

Call: desc_clients_status
Argument Switch: None
Argument: None
Allowed Pre-License: No
Purpose: Provides information on connected overlay network devices (sometimes referred to as OLNDs)

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address desc_ipsec_status

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.73 desc_ipsec_status

Response:

```
connected_clients:
  172.31.1.53:
    ipaddress: 10.160.139.146
    managerid: 3
    overlay_ipaddress: 172.31.1.53
 <more>
```

Call: desc_system_status
Argument Switch: None
Argument: None
Allowed Pre-License: No
Purpose: Provides information about the underlying appliance; memory, cpu, disk space, etc..

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address desc_system_status

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.73 desc_system_status

Response:
TBD

Call: desc_link_history
Argument Switch: --cidr (CIDR)
Argument: IP CIDR description of the subnet information is desired for
Allowed Pre-License: No
Purpose: Provides information about the connection history of the subnet's IPsec connection

command prompt> ruby vpncubed.rb -K "api" -S "myapisecret" -H manager-ip-address desc_link_history --cidr cidr_as_string

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.73 --cidr "192.168.1.0/24"

Response:
TBD

VPN3 API: License

Call: upload_license
Argument Switch: --license
Argument: Valid path to a file containing the encrypted license
Allowed Pre-License: Yes
Purpose: License a VPN3 Manager to be a part of a specific topology

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address upload_license --license /pathtolicensefile/license.txt

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.73 upload_license --license apidemo_ipsectrial_license.txt

Response:
license: accepted

Note: This call is only available for SME and Enterprise Editions. Free and Lite Editions come preconfigured with the appropriate license.

Call: desc_license
Argument Switch: None
Argument: None
Allowed Pre-License: No
Purpose: Provides information about the manager's license

command prompt> ruby vpncubed.rb -K "api" -S "myapisecret" -H manager-ip-address desc_license

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.73 desc_license

Response:

(see following page)

VPN3 API: License

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.73 desc_license

```
capabilities:
- IPsec
topology:
  managers:
  - manager_id: 1
    overlay_ipaddress: 172.31.1.1
  clients:
  - 172.31.1.5
  - 172.31.1.9
  - 172.31.1.13
  - 172.31.1.17
  - 172.31.1.21
uploaded_at: Sat Aug 28 15:48:31 +0000 2010
sha1_checksum: 5f700d95c0e836ecf0cc9996cf6970661136e5b1
uploaded_at_i: 1283010511
license_present: true
```

Call: desc_keyset
Argument Switch: None
Argument: None
Allowed Pre-License: No
Purpose: Returns status of whether cryptographic credentials, which are used to provide overlay devices access to the topology, have been generated.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address desc_keyset

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.73 desc_keyset

Response: (One of three states; there are no keys, they are being generated, they have been generated.)

```
keyset_present: false
in_progress: false
```

```
keyset_present: false
running: 0
in_progress: true
```

```
keyset_present: true
created_at: 2010/08/28 19:25:57 +0000
created_at_i: 1283023557
checksum: 0d61536900908f1b985eae65aa9473a2f312c94c
```

Call: setup_keyset
Argument Switch: --source, --token
Argument: Source Manager with hostname or address as a string, Token is any arbitrary key used to help randomize the checksum, it must be identical for each manager in a topology.
Allowed Pre-License: No
Purpose: Generates cryptographic credentials which are used to provide overlay devices access to the topology.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H another-manager-ip-address setup_keyset --source manager-ip-address --token "arandomstring”

Note: *--source is not required for the first invocation on a manager in the topology, without --source the keys are generated, with source the keys are pulled from one manager to another.*

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.73 setup_keyset --token “arandomstring”

Response:

(see following page)

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.73 setup_keyset --token “arandomstring”

Response:
keyset_present: false
running: 0
in_progress: true
started_at_i: 1283023530
started_at: 2010/08/28 19:25:30 +0000

After several minutes; query again, and the keyset has been generated.

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.73 desc_keyset

Response:
keyset_present: true
created_at: 2010/08/28 19:25:57 +0000
created_at_i: 1283023557
checksum: 0d61536900908f1b985eae65aa9473a2f312c94c

Now invoke the command on ANOTHER overlay manager; and get the keys.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H another-manager-ip-address setup_keyset --source manager-ip-address --token "arandomstring”

NOTE: “-H” is a different machine in this example with our previous host as the --source.

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.68 setup_keyset --token “arandomstring” --source 174.129.238.73

Response:
keyset_present: false
running: 1
in_progress: true
started_at_i: 1283114203
started_at: 2010/08/29 20:36:43 +0000

Note: You will see that the hosts from our example both have the same checksum when a query of “desc_keyset” id performed. This means the 2 managers can be peered together in the topology.

Call: desc_clientpacks
Argument Switch: None
Argument: None
Allowed Pre-License: No
Purpose: Returns detailed information about all of the clientpacks in the topology. If manager's are properly peered, this information can come from any of the managers.

command prompt> ruby vpncubed.rb -K "api" -S "myapisecret" -H manager1-ip-address desc_clientpacks

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.68 desc_clientpacks

Response: (a list of this information for all clientpacks in the topology)

```
172.31.1.17:
  name: 172_31_117
  tarball_file: 172_31_117.tar.gz
  tarball_sha1: 4e81448988b2c72bedc8349cbb0c8ccc5541b457
  checked_out: false
  zip_file: 172_31_1_17.zip
  zip_sha1: 2078a82cd1939037f548bfea3739783ffe0e8b6a
  enabled: true
  sequential_id: 1
  overlay_ipaddress: 172.31.1.17
```

Call: fetch_clientpack
Argument Switch: --name, --format, -o
Argument: The “--name” is the name of the clientpack as returned by the “desc_clientpacks” call. The “ --format” has two possible values “tarball” (default) or “zip” which as stated, returns the clientpack in that compression format. The “-o” is an output file name for the clientpack
Allowed Pre-License: No
Purpose: Clientpacks are compressed files with the necessary information and credentials for an overlay client to be connected to the VPN-Cubed topology.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager1-ip-address fetch_clientpack --name 172_31_1_17 --format “zip” -o “mycredentials.zip”

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.68 fetch_clientpack --name “172_31_1_17” --format “zip” -o “mycredentials.zip”

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.68 fetch_clientpack --name “172_31_1_17” --format “tarball” -o “mycredentials.tar”

Response: No success code is printed. Clientpack is output to the requsted file name.

Call: get_next_available_clientpack (see warning on following page)
Argument Switch: None
Argument: None
Allowed Pre-License: No
Purpose: For situations where devices need overlay credentials but not for a specific address. This is usually in “autoscale” situations.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager1-ip-address get_next_available_clientpack

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.68 get_next_available_clientpack

Response: (the info of the next available clientpack for use in a subsequent “fetch_clientpack”)

```
name: 172_31_1_17
tarball_file: 172_31_1_17.tar.gz
tarball_sha1: 3c6212414a27a1507fe518038495deeb4b8a382c
checked_out: true
zip_file: 172_31_1_17.zip
zip_sha1: d1590336eb5540d27913d8bff9057a8b36fef8b4
enabled: true
sequential_id: 1
overlay_ipaddress: 172.31.1.17
```

Warning: You must use only one manager of the topology for use with “get_next_available_clientpack”! VPN3 Managers are not (at this time) able to synchronize information automatically. Using multiple managers will cause distribution of the same client pack to multiple overlay devices which is not allowed. (Multiple clients with the same credentials will continuously knock each other off of the overlay network.)

Note: The “get_next_available_clientpack” call does not actually fetch the credentials. It provides sufficient information for you to the call “fetch_clientpack”.

Note: In order for get_next_available_clientpack to not exhaust your pool of clientpacks, devices need to release the clientpack when done by calling “edit_clientpack” for their clientpack, setting the “--checked_out” property to “false”.

VPN3 API: Clientpacks

Call: edit_clientpack
Argument Switch: --name, --enabled, --checked_out
Argument: The “--name” switch is the name of a clientpack, Both “--enabled” and “--checked_out” take “true” or “false” as values.
Allowed Pre-License: No
Purpose: For changing properties of clientpacks.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager1-ip-address --name “172_31_1_17” --enabled true --checked_out false

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.68 --name “172_31_1_17” --enabled true --check_out false

Response: (returns the clientpack info for the named clientpack)
name: 172_31_1_17
tarball_file: 172_31_1_17.tar.gz
tarball_sha1: 3c6212414a27a1507fe518038495deeb4b8a382c
checked_out: false
zip_file: 172_31_1_17.zip
zip_sha1: d1590336eb5540d27913d8bff9057a8b36fef8b4
enabled: true
sequential_id: 1
overlay_ipaddress: 172.31.1.17

Call: create_snapshot
Argument Switch: None
Argument: None
Allowed Pre-License: No
Purpose: Create a snapshot file on that manager being queried. The snapshot is named based on date, time and IP address of the manager.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address create_snapshot

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.68 create_snapshot

Response:
snapshot_20100830_1283176087_174.129.238.68:
size: 912974
created_at: 2010/08/30 13:48:08 +0000
created_at_i: 1283176088
sha1_checksum: 6b1ddca58d454022ca804c31fc016447613df218

Note: Snapshots have a 1-to-1 relationship with managers in a topology. To recover an “N” manager topology from snapshots requires “N” distinct snapshot files.

Call: fetch_snapshot
Argument Switch: --name, -o
Argument: The “--name” switch is the name of the snapshot desired. “The “-o” switch is for the name of the output file you want for the snapshot.
Allowed Pre-License: No
Purpose: Download a snapshot file for later use.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address fetch_snapshot --name snapshot_as_string -o mysnapshotfile

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.68 fetch_snapshot --name snapshot_20100830_1283176087_174.129.238.68 -o m1_snapshot

Response: The requested snapshot file is downloaded to the file specified by the “-o” switch.

Call: delete_snapshot
Argument Switch: --name
Argument: The “--name” switch is the name of the snapshot desired.
Allowed Pre-License: No
Purpose: Delete the named snapshot from the manager.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address delete_snapshot --name snapshot_as_string

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.68 delete_snapshot --name snapshot_20100830_1283176087_174.129.238.68

Response: (the call returns the list of remaining snapshots, in this case an empty list)
{}

Call: import_snapshot
Argument Switch: --snapshot
Argument: The “--snapshot” switch is the file containing the snapshot you wish to import.
Allowed Pre-License: Yes
Purpose: Imports the snapshot into the manager and triggers a reboot for the configuration to take effect.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address import_snapshot --snapshot filename_of_snapshot

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.68 import_snapshot --snapshot m1_snapshot

Response:
snapshot: accepted

Call: desc_snapshots
Argument Switch: None
Argument: None
Allowed Pre-License: No
Purpose: Shows snapshots that have been taken on that manager being queried.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address desc_snapshots

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.68 desc_snapshots

Response:

```
snapshots:
  snapshot_20100830_1283176087_174.129.238.68:
    size: 912974
    created_at: 2010/08/30 13:48:08 +0000
    created_at_i: 1283176088
    sha1_checksum: 6b1ddca58d454022ca804c31fc016447613df218
latest_snapshot: snapshot_20100830_1283176087_174.129.238.68
```

Note: This response shows either an empty list token “{}” or returns a list of snapshots with the token “latest_snapshot” indicating the most recent snapshot created.

VPN3 API: Manager Peering

Peering Managers is required in all Editions. The Free and Lite Edition allow for only single Manager topologies but the single Manager must be setup up as Manager 1 in order the overlay network to function.

For Free and Lite Editions only run the “set_manager_id” as shown below.

cmd prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address set_manager_id --id 1
ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.68 set_manager_id --id 1

Call: desc_peering
Argument Switch: None
Argument: None
Allowed Pre-License: No
Purpose: Provides the status of whether a manager is peered to other managers.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address desc_peering

Note: In the order of operations performed, there are 2 managers; at 174.129.238.68 and 174.129.238.73. Neither of them has been peered so this operation will return false until add_peer has been called on them. Prior to add_peer the managers must be given “IDs” for their manager number in the topology via the set_manager_id call.

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.73 desc_peering

Response:
false

Call: set_manager_id
Argument Switch: --id
Argument: The manager ID as an integer, cannot be the same as the id of another manager in the topology, and cannot be greater than the number of managers in the topology.
Allowed Pre-License: No
Purpose: Sets the Manager ID of a manager so that it can be peered within a topology.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address set_manager_id --id 1

Note: We will set the manager ID of our example host 174.129.238.68 to “1” and our example host 174.129.238.73 to “2”. They will then have the same license, keysets, checksum, and proper manager ids, meaning they can then subsequently be “peered”.

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.68 set_manager_id --id 1

Response:
id: 1
managers:
"1":
self: true
id: 1

VPN3 API: Manager Peering

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.73 set_manager_id --id 2

Response:

```
peered: true
id: 2
managers:
  "1":
    not_set: true
    id: 1
  "2":
    self: true
    id: 2
  "3":
    not_set: true
    id: 3
  "4":
    not_set: true
    id: 4
```

Now that Manager ID's are set, add_peer commands can be done.

Call: add_peer
Argument Switch: --id, --name
Argument: The “--id” is the manager ID as an integer of the the manager you are peering with, NOT the id of the manager you are calling “add_peer” on. The “--name” is the IP address or host name of the manager you are peering with.
Allowed Pre-License: No
Purpose: Creates a peering relationship from a manager to another manager. The peering call is unidirectional. Reciprocal calls must be made to peer two managers together.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager1-ip-address add_peer --id 2 --name manager2-ip-address

Note: The manager ID of our example host 174.129.238.68 is “1” and our example host 174.129.238.73 is “2”.

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.68 add_peer --id 2 --name 174.129.238.73

Response: (next page)

Response:

```
peered: true
id: 1
managers:
  "1":
    self: true
    id: 1
  "2":
    address:174.129.238.73
    reachable: true
    id: 2
  "3":
    not_set: true
    id: 3
 "4":
   not_set: true
    id: 4
```

Now we can do reciprocal manager peering calls to establish the mesh connection between the managers.

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.73 add_peer --id 1 --name 174.129.238.68

NOTE: Sometimes “reachable” will say :false for a short period of time before the managers finish their connection process. If it says :false for a long period of time, there is an error.

Call: delete_peer
Argument Switch: --id
Argument: The “--id” is the manager ID as an integer of the the manager you want to break your peering relationship with.
Allowed Pre-License: No
Purpose: Breaks a peering relationship from a manager to another manager. The peering call is unidirectional. Reciprocal calls must be made to fully break the peer relationship.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager1-ip-address delete_peer --id 2

Note: The manager ID of our example host 174.129.238.68 is “1” and our example host 174.129.238.73 is “2”. When the delete_peer call is made we are returned to the state prior to the add_peer call.

ruby $vpn3api_home/vpncubed.rb -K api -S apidemopassword -H 174.129.238.68 delete_peer --id 2

Response:

```
peered: true
id: 1
managers:
  "1":
    self: true
    id: 1
  "2":
    not_set: true
    id: 2....
```

Note: IPsec configuration from the command line is the most complicated of the calling sequences covered in this document. It requires knowledge of IPsec connectivity.

In order to show the commands a pre-existing network configuration will be connected to via IPsec. We have used an IPSec device running at 184.72.229.148. It is running with NAT-Traversal enabled and uses the preshared key “vpncubedrocks”. It has Perfect Forward Secrecy Enabled and is looking for “aes256” encryption for both Phase 1 and Phase 2 negotiations. It is looking for “sha1” hash method for Phase 1 and Phase 2 negotiations. It uses Diffie Hellman Group 5 when needed.

Our approach with the API will be:
- Create an IPsec endpoint and then interrogate it and manipulate it with the other calls.

Call: create_ipsec_endpoint
Argument Switch: --name (string), --ipaddress (ip address xxx.xxx.xxx.xxx), --secret (string), --pfs (true or false), --extra_config (string), --cloud_wan_subnets, --private_ipaddress (ip address xxx.xxx.xxx.xxx)
Argument: The "--name" arg is a user-supplied name for the connection which will show up as a label in the web UI. The "--ipaddress" arg is the IP Address of the remote gateway. The "--secret" arg is the pre-shared key for the connection. The "--pfs" enables Perfect Forward Secrecy if true, disables if false. The "--extra_config" arg is a string with additional options for the connection, the string must include line breaks as displayed below. The "--private_ipaddress" arg is the internal NAT address of the remote gateway.
Allowed Pre-License: No
Purpose: Create IPsec connection to the defined remote gateway.

command prompt> ruby vpncubed.rb -K "api" -S "myapisecret" -H manager-ip-address create_ipsec_endpoint --name "IPsec Connection" --ipaddress remote-gateway-address --secret preshared-key --pfs true --extra_config

ruby $vpn3api_home/vpncubed.rb -K api -S "apidemopassword" -H 174.129.238.73 create_ipsec_endpoint --name "API_Created_Entry_for_Remote_System" --ipaddress 184.72.229.148 --secret "vpncubedrocks" --pfs true --extra_config "ike=aes256-sha1
esp=aes256-sha1
dpdaction=restart
dpdtimeout=120
dpddelay=30"

Call: create_ipsec_endpoint (continued)

Response:
name: API_Created_Entry_for_Remote_System
ipaddress: 184.72.229.148
id: 1
extra_config:
- ike=aes256-sha1
- esp=aes256-sha1
- dpdaction=restart
- dpdtimeout=120
- dpddelay=30
cloud_wan_subnets: []
pfs: true
remote_subnets: {}
bgp_peers: {}

Note: This defined the remote gateway endpoint. In order to create a live tunnel a subnet definition needs to be added to this endpoint.

Call: desc_ipsec
Argument Switch: None
Argument: None
Allowed Pre-License: No
Purpose: Returns information about all IPsec endpoints and subnets defined.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address desc_ipsec

ruby $vpn3api_home/vpncubed.rb -K api -S "apidemopassword" -H 174.129.238.73 desc_ipsec

Response: (next page)

Call: desc_ipsec (continued)

Response:

```
this_endpoint:
  ipaddress: 174.129.238.73
  asn: 65002
  private_ipaddress: 192.0.2.254
  local_subnet: 172.31.1.0/24
remote_endpoints:
  "1":
    name: API_Created_Entry_for_Remote_System
    ipaddress: 184.72.229.148
    id: 1
    extra_config:
    - ike=aes256-sha1
    - esp=aes256-sha1
    - dpdaction=restart
    - dpdtimeout=120
    - dpddelay=30
cloud_wan_subnets: []
pfs: true
remote_subnets: {}
bgp_peers: {}
```

Call: setup_ipsec
Argument Switch: --restart (true or false)
Argument: The "--restart" arg restarts the VPN3 manager's IPsec subsystem when the value is "true". The value of "false" has no value.
Allowed Pre-License: No
Purpose:

command prompt> ruby vpncubed.rb -K "api" -S "myapisecret" -H manager-ip-address setup_ipsec --restart true

ruby $vpn3api_home/vpncubed.rb -K api -S "apidemopassword" -H 174.129.238.73 setup_ipsec --restart true

Response:
restart: true

Call: get_ipsec_local_ipaddress
Argument Switch: None
Argument: None
Allowed Pre-License: No
Purpose: Because VPN3 managers run as virtual instances in virtual infrastructure - there is a possibility that an instance will be re-launched, ending up with a different internal NAT address than it originally had, which would require all connected devices to change their information. As a result VPN3 has a “fake” NAT which is used so that it can remain constant across launches, meaning no reconfiguration by peer gateways.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address get_ipsec_local_ipaddress

ruby $vpn3api_home/vpncubed.rb -K api -S "apidemopassword" -H 174.129.238.73 get_ipsec_local_address

Response:
ipsec_local_ipaddress: 192.0.2.254

Call: set_ipsec_local_ipaddress
Argument Switch: None
Argument: None
Allowed Pre-License: No
Purpose: Because VPN3 managers run as virtual instances in virtual infrastructure - there is a possibility that an instance will be re-launched, ending up with a different internal NAT address than it originally had, which would require all connected devices to change their information. As a result VPN3 has a “fake” NAT which is used so that it can remain constant across launches, meaning no reconfiguration by peer gateways.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address set_ipsec_local_ipaddress --ipaddress my_new_persistent_NAT_address

ruby $vpn3api_home/vpncubed.rb -K api -S "apidemopassword" -H 174.129.238.73 set_ipsec_local_address --ipaddress 192.0.8.200

Response:
Error: OperationNotAllowedError: Detected active IPsec configuration.

Note: This call will not work if there are any defined IPsec subnets or Remote Endpoints. In this case since we defined a remote endpoint; an error it returned.

Call: edit_ipsec_endpoint
Argument Switch: --endpointid (integer), --name (string), --ipaddress (ip address xxx.xxx.xxx.xxx), --secret (string), --pfs (true or false), --extra_config (string), --cloud_wan_subnets, --private_ipaddress (ip address xxx.xxx.xxx.xxx)
Argument: The “--endpointid” arg is the id number of the specific endpoint which is being edited. The “--name” arg is a user-supplied name for the connection which will show up as a label in the web UI. The “--ipaddress” arg is the IP Address of the remote gateway. The “--secret” arg is the pre-shared key for the connection. The “--pfs” enables Perfect Forward Secrecy if true, disables if false. The “--extra_config” arg is a string with additional options for the connection, the string must include line breaks as displayed below. The “--private_ipaddress” arg is the internal NAT address of the remote gateway.
Allowed Pre-License: No
Purpose: Edit IPsec connection to the defined remote gateway.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address edit_ipsec_endpoint --endpointid 1 --private_ipaddress gateway_nat_address

ruby $vpn3api_home/vpncubed.rb -K api -S "apidemopassword" -H 174.129.238.73 create_ipsec_endpoint --endpointid 1 --private_ipaddress 192.168

Response: (next page)

Call: edit_ipsec_endpoint (continued)
Response:

```
name: API_Created_Entry_for_Remote_System
ipaddress: 184.72.229.148
id: 4
extra_config:
- ike=aes256-sha1
- esp=aes256-sha1
- dpdaction=restart
- dpdtimeout=120
- dpddelay=30
cloud_wan_subnets: []
pfs: true
private_ipaddress: 192.0.4.250
remote_subnets: {}
bgp_peers: {}
```

Note: In this example we used edit_ipsec_endpoint to add a property we didn't set the first time; the NAT address of the remote gateway.

Call: desc_ipsec_endpoint
Argument Switch: --endpointid
Argument: The “--endpointid” arg specifies which of the defined remote endpoints for which detailed information is desired.
Allowed Pre-License: No
Purpose: Returns detailed information about the IPsec endpoint specified.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address desc_ipsec_endpoint --endpointid integer_of_one_of_the_endpoints

ruby $vpn3api_home/vpncubed.rb -K api -S "apidemopassword" -H 174.129.238.73 desc_ipsec_endpoint --endpointid 1

Response:
The same information is returned for this single IPsec endpoint specified - identical to the information returned via desc_ipsec.

Call: create_remote_subnet
Argument Switch: --endpointid (integer), --subnet (CIDR)
Argument: The “--endpointid” arg specifies the remote endpoints for which a subnet tunnel is to be created. The subnet is a CIDR specification for the subnet at the remote gateway.
Allowed Pre-License: No
Purpose: Creates an IPsec tunnel to the remote gateway for communicating with the subnet specified.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address create_remote_subnet --endpointid integer_of_one_of_the_endpoints --subnet cidr_spec

ruby $vpn3api_home/vpncubed.rb -K api -S "apidemopassword" -H 174.129.238.73 create_remote_subnet --endpointid 1 --subnet 192.168.1.0/24

Response:

```
remote_subnets:
  "2":
    id: 2
    subnet: 192.168.1.0/24
```

Note: The endpoint info is returned along with a “remote_subnets:” section. The endpoint IDs and remote subnets share the same numbering sequence. In this case with the endpoint having an ID of “1”, the next endpoint or subnet gets an id of “2”, as seen in this response.

Call: delete_remote_subnet
Argument Switch: --endpointid (integer), --subnetid (integer)
Argument: The “--endpointid” arg specifies the remote endpoint for which a subnet tunnel is to be deleted. The “--subnetid” arg specifies a subnet which must be associated with the endpoint chosen.
Allowed Pre-License: No
Purpose: Removes an IPsec tunnel to the remote gateway as specified by the subnet id.

command prompt> ruby vpncubed.rb -K “api” -S “myapisecret” -H manager-ip-address
delete_remote_subnet --endpointid integer_of_one_of_the_endpoints --subnetid subnet_id

ruby $vpn3api_home/vpncubed.rb -K api -S "apidemopassword" -H 174.129.238.73 delete_remote_subnet
--endpointid 1 --subnetid 2

Response:
The endpoint info is returned.

Issue appears to be "hopping" on and off the network. This is usually the result of the same client keys being installed on two client machines in the network. Only one client machine can use a set of credentials at a given time.

Fetch Keyset appears to hang or not work. Check to see if the Amazon security group is correct for port 8000 between the manager you are getting the keyset from and the manager you are do the fetch from. If they are separated across Amazon USA and Amazon EU you will need to have thier security group reference the public IP addresses. When you do the "Fetch Keyset" command use the managers public IP address.

Manager IDs seem correct, EC2 security groups seem correct, but managers, especially ones launched via separate launch commands will not "peer". Review your worksheet and your launch commands. Ensure that the managers were all launched with the same alphanumeric string.

End