MetroAE Config Documentation

MetroAE Config - Overview

metroae config is command-line-driven tool that allows you to create and delete VSD configurations. It utilizes the VSD API along with a set of common Feature Templates to create configurations in the VSD. You will create a yaml or json data file with the necessary configuration parameters for the particular feature and execute a simple CLI command *"metroae config create"* to push the configuration into VSD. You can get more information about usage by referring to MetroAE Config Usage.

How is MetroAE Config delivered?

MetroAE Config is part of the wider MetroAE platform. However, it is only supported in the container-based version of MetroAE. All necessary components are provided within the container and in part during the metroae config setup process. You can get more information about installation by referring to Installation and Setup.

What is a Feature Template?

A feature template is a provided yaml file that defines the data required by the VSD along with API interaction to create specific configuration in the VSD. MetroAE Config provides a set of supported Feature Templates that cover specific feature configuration in the VSD. Feature Templates do not need to have a 1 to 1 relationship with a VSD object, these can be abstracted to simplify the user data requirements. You can get more information about Feature Templates by referring to Description of a Feature Template.

Who provides the Feature Templates?

MetroAE Config provides you with a supported set of Feature Templates. The list of supported, standard Feature Templates will

grow over time to cover more portions of VSD configuration along with greater work flow, user data and feature abstraction. The Feature Templates are authored and tested by the Nuage Automation team. It is the intention to also provide experimental and community authored templates as we grow the user community for the tool.

How is it different than VSPK?

The Nuage VSPK is a set of libraries provided in multiple frameworks that allow a user to interact with the VSD API. The frameworks supported are common tools sets like python, ansible and Go. To utilize the VSPK the user will have to understand the necessary API objects, calls and relationships and subsequently script (in the chosen language) the configuration work flow that is required. VSPK is a powerful tool and is often used to integrate the Nuage VSD into north bound OSS and configuration portals etc.

However, for users who are not well versed in one of the frameworks supported and may not be proficient at programming/scripting there is still a gap. Of course, the VSD provides a rich UI. However, for repeated configurations and complex network designs, the UI is not very efficient and can be prone to human error, particularly when dealing with large configurations or advanced features.

MetroAE Config takes a different approach to the configuration challenge. Rather than exposing a library and having the user develop code, a template driven approach to configuration is provided. Templates are authored to interact directly with the VSD API, abstracting the API calls, objects and relationships away from the user. You simply provide the required “data” for a particular feature and the MetroAE Config engine handles all the API interaction.

MetroAE Config Engine Installation

MetroAE Configuration engine is provided as a Docker container. You can install the container and the supporting artifacts using the `metroae` srcript. By providing a few simple inputs, you can automatically install the Config Engine along with its dependencies.

On a host where the configuration engine will be installed the following artifacts will be installed by `metroae`:

- Docker container for configuration
- Collection of configuration Templates
- VSD API Specification

System Requirements

The MetroAE container requires the following on the host where it will be installed:

- Operating System: RHEL/Centos 7.4+
- Docker: Docker Engine 1.13.1+
- Network Access: Internal and Public
- Storage: At least 800MB

Operating system

The primary requirement for the configuration container is Docker Engine, however the installation, container operation and functionality is only tested on RHEL/Centos 7.4. You can use one of the many other operating systems will support Docker Engine, however support for these is not currently provided and would be

considered experimental only. A manual set of installation steps is provided for you to use in these cases.

Docker Engine

The configuration engine is packaged into a Docker container. This ensures that all package and library requirements are self-contained with no other host dependencies. To support this, Docker Engine must be installed on the host. The configuration container requirements, however, are quite minimal. Primarily, the Docker container mounts a local path on the host as a volume while ensuring that any templates and user data are maintained only on the host. The user never needs to interact directly with the container.

Network Access

Currently the MetroAE container can be pulled from either an internal Nokia Docker registry or an Amazon AWS S3 bucket on the public network. The Feature Templates and VSD API Specifications are hosted publicly, as well. The `metroae` install script manages the location of these resources automatically. You don't need to manage this or provide any further information. However, public network access during the installation is required.

Storage

The MetroAE container along with the templates requires approximately 800MB of local disk space. Note that the container itself is ~750MB, thus it is recommended that during install a good network connection is available.

User Privileges

The user must have elevated privileges on the host system.

Installing via installation script

1. Install Docker Engine

```
[metroae-user@metroae-host]# yum install docker -y
Loaded plugins: search-disabled-repos
Resolving Dependencies
--> Running transaction check
---> Package docker.x86_64 2:1.13.1-88.07f3374.el7 will be
installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================

 Package                          Arch
Version
Repository                                                  Size
================================================================

Installing:
 docker                           x86_64
2:1.13.1-88.git07f3374.el7
rhel-7-server-extra-rpms                                    16 M

Transaction Summary
================================================================

Install  1 Package

Total download size: 16 M
Installed size: 57 M
Downloading packages:
docker-1.13.1-88.git07f3374.el7.x86_64.rpm
|  16 MB  00:00:03
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : 2:docker-1.13.1-88.git07f3374.el7.x86_64
1/1
  Verifying  : 2:docker-1.13.1-88.git07f3374.el7.x86_64
1/1

Installed:
  docker.x86_64 2:1.13.1-88.git07f3374.el7

Complete!
```

2. Enable and Start Docker Engine Service

```
[metroae-user@metroae-host]# systemctl enable docker
Created symlink from /etc/systemd/system/multi-
user.target.wants/docker.service to
/usr/lib/systemd/system/docker.service.
[metroae-user@metroae-host]# systemctl start docker
```

Verify that the service is running.

```
[root@metroae-host ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service;
enabled; vendor preset: disabled)
   Active: active (running) since Tue 2019-07-09 16:41:14
UTC; 3min 41s ago
     Docs: http://docs.docker.com
 Main PID: 3206 (dockerd-current)
   CGroup: /system.slice/docker.service
           ├─3206 /usr/bin/dockerd-current --add-runtime
docker-runc=/usr/libexec/docker/docker-runc-current --
default-runtime=docker-runc --exec-opt
native.cgroupdriver=systemd --use...
           └─3220 /usr/bin/docker-containerd-current -l
unix:///var/run/docker/libcontainerd/docker-
containerd.sock --metrics-interval=0 --start-timeout 2m --
state-dir /var/run/docker...
```

We can check wether docker is running also via running a sample docker command. ie. “docker version”

```
[root@metroae-host ~]# docker --version
Docker version 1.13.1, build 07f3374/1.13.1
```

3. Add the user to the wheel and docker groups. In this case the user is “caso”

```
[root@metroae-host ~]# gpasswd -a caso wheel
Adding user caso to group wheel
[root@metroae-host ~]# gpasswd -a caso docker
Adding user caso to group docker
```

4. Move or Copy the “metroae” script to /usr/bin and set permissions correctly to make the script executable.

```
[root@metroae-host ~]# mv metroae /usr/bin
[root@metroae-host ~]# chmod 755 /usr/bin/metroae
[root@metroae-host ~]# ls -la /usr/bin | grep metroae
-rwxr-xr-x.  1 root root      64264 Jul  9 17:13 metroae
```

5. Switch to the user that will operate “metroae config” and check the script is running ok. In this case per step 3 its the “caso” user.

```
[caso@metroae-host ~]$ metroae help
Nuage Networks Metro Automation Engine (MetroAE) Version:
v3.3.0

MetroAE usage:

metroae config
Configure VSD
metroae config help
Displays the help text for levistate
metroae config version
Displays the current levistate version
metroae config engine update
Update levistate to the latest version
metroae config <positional args>
Execute levistate inside the container
metroae container
Manage the MetroAE container
metroae container pull
Pull a new MetroAE image from the registry
metroae container setup
Setup the mertroae container
metroae container start
Start the mertroae container
metroae container stop
Stop the mertroae container
metroae container status
Display the status of the mertroae container
metroae container destroy
Destroy the mertroae container
metroae container update
Update the mertroae container to the latest version
metroae container ssh copyid
Copy the container ssh public key to a host

Additional menu help is available by adding 'help' to the
command line,
e.g. 'metroae container help'
```

Note if previously the user was not added to wheel and docker there would be various permissions issues seen here.

6. Setup the container using the metroae script.

   We are going to pull the image and setup the metro container in one command below. During the install we will be prompted for a location for the container data directory. This is the location where our user data, templates and VSD API specs will be installed and created and a volume mount for the container will be created. However this can occur orthogonally via “pull” and “setup” running at separate times which can be useful depending on available network bandwidth.

```
[caso@metroae-host ~]$ metroae container help
Nuage Networks Metro Automation Engine (MetroAE) Version:
v3.3.0

MetroAE container usage:

metroae container
Manage the MetroAE container
metroae container pull
Pull a new MetroAE image from the registry
metroae container setup
Setup the mertroae container
metroae container start
Start the mertroae container
metroae container stop
Stop the mertroae container
metroae container status
Display the status of the mertroae container
metroae container destroy
Destroy the mertroae container
metroae container update
Update the mertroae container to the latest version
metroae container ssh copyid
Copy the container ssh public key to a host
```

   We will use "metroae container setup".

```
[caso@metroae-host ~]$ metroae container setup

>>> Checking docker version

>>> Setup MetroAE container
```

```
Elevated privileges are required for setup. During setup,
we may prompt you
for the sudo password.

>>> Pulling the MetroAE container image from the
repository


>>> Pulling the MetroAE container image from Nokia
registry

Trying to pull repository
registry.mv.nuagenetworks.net:5000/metroae ...
softlaunch: Pulling from
registry.mv.nuagenetworks.net:5000/metroae
9a598663ae73: Pulling fs layer
b827d31025a3: Pulling fs layer
97bbe95d2d2f: Pulling fs layer
37ac171a5273: Pulling fs layer
d1708ac1780b: Pulling fs layer
3d34009f3311: Pulling fs layer
8a68a0cc2542: Pulling fs layer
3a8eefbe7d69: Pulling fs layer
0e16004b6d62: Pulling fs layer
15c94733df9e: Pulling fs layer
57ed50c82d62: Pulling fs layer
685cde86f35f: Pulling fs layer
f59cd4a686f5: Pulling fs layer
8903b9128b3d: Pulling fs layer
8f0aca9b1856: Pulling fs layer
700a872398e6: Pulling fs layer
6dfae24ca868: Pulling fs layer
df6b49f8cc7d: Pulling fs layer
4e368f70cd0b: Pulling fs layer
37ac171a5273: Waiting
d1708ac1780b: Waiting
3d34009f3311: Waiting
8a68a0cc2542: Waiting
3a8eefbe7d69: Waiting
0e16004b6d62: Waiting
15c94733df9e: Waiting
57ed50c82d62: Waiting
685cde86f35f: Waiting
f59cd4a686f5: Waiting
8903b9128b3d: Waiting
8f0aca9b1856: Waiting
700a872398e6: Waiting
6dfae24ca868: Waiting
df6b49f8cc7d: Waiting
4e368f70cd0b: Waiting
b827d31025a3: Verifying Checksum
b827d31025a3: Download complete
37ac171a5273: Verifying Checksum
37ac171a5273: Download complete
9a598663ae73: Verifying Checksum
9a598663ae73: Download complete
```

```
97bbe95d2d2f: Verifying Checksum
97bbe95d2d2f: Download complete
8a68a0cc2542: Verifying Checksum
8a68a0cc2542: Download complete
d1708ac1780b: Verifying Checksum
d1708ac1780b: Download complete
3a8eefbe7d69: Verifying Checksum
3a8eefbe7d69: Download complete
15c94733df9e: Verifying Checksum
15c94733df9e: Download complete
57ed50c82d62: Verifying Checksum
57ed50c82d62: Download complete
9a598663ae73: Pull complete
b827d31025a3: Pull complete
3d34009f3311: Verifying Checksum
3d34009f3311: Download complete
f59cd4a686f5: Verifying Checksum
f59cd4a686f5: Download complete
8903b9128b3d: Verifying Checksum
8903b9128b3d: Download complete
8f0aca9b1856: Verifying Checksum
8f0aca9b1856: Download complete
685cde86f35f: Verifying Checksum
685cde86f35f: Download complete
700a872398e6: Verifying Checksum
700a872398e6: Download complete
0e16004b6d62: Verifying Checksum
0e16004b6d62: Download complete
df6b49f8cc7d: Verifying Checksum
df6b49f8cc7d: Download complete
4e368f70cd0b: Verifying Checksum
4e368f70cd0b: Download complete
6dfae24ca868: Verifying Checksum
6dfae24ca868: Download complete
97bbe95d2d2f: Pull complete
37ac171a5273: Pull complete
d1708ac1780b: Pull complete
3d34009f3311: Pull complete
8a68a0cc2542: Pull complete
3a8eefbe7d69: Pull complete
0e16004b6d62: Pull complete
15c94733df9e: Pull complete
57ed50c82d62: Pull complete
685cde86f35f: Pull complete
f59cd4a686f5: Pull complete
8903b9128b3d: Pull complete
8f0aca9b1856: Pull complete
700a872398e6: Pull complete
6dfae24ca868: Pull complete
df6b49f8cc7d: Pull complete
4e368f70cd0b: Pull complete
Digest:
sha256:422fcd0ffe7b02c910c2824c4a074575c5fb2d5aed8d2fca8d3b595

Status: Downloaded newer image for
registry.mv.nuagenetworks.net:5000/metroae:softlaunch
Successfully Pulled the MetroAE container image from
```

```
Docker registry

>>> Setup container for MetroAE config only

Data directory configuration

The MetroAE container needs access to your user data. It
gets access by internally
mounting a directory from the host. We refer to this as
the 'data directory'.
The data directory is where you will have deployments,
templates, documentation,
and other useful files.

Please specify the full path to the data directory on the
Docker host. Setup will
make sure that the path ends with 'metroae_data'. If the
path you specify does
not end with 'metroae_data', setup willl create it.

Data directory path: /home/caso/

Checking path: Data directory path
>>> Data directory path set to: /home/caso/metroae_data


>>> Starting the MetroAE container

574c7314aa7ee61ab6d29effa76dc233409ccf1a3aad3328174ac86b691a5b

MetroAE container started successfully

>>> Pulling the latest templates and files for MetroAE
config in the container

Updating templates...

MetroAE container setup complete. Execute 'metroae
container status' for status.


[MetroAE v3.3.0, script 1.0.4, container softlaunch]
```

At this point the container should be running and the templates and VSD API spec should be installed in the specified data directory.

```
[caso@metroae-host ~]$ docker ps
CONTAINER ID        IMAGE
COMMAND                  CREATED             STATUS
PORTS               NAMES
574c7314aa7e
registry.mv.nuagenetworks.net:5000/metroae:softlaunch
```

```
"/bin/sh -c /sourc..."   2 minutes ago       Up 2 minutes
metroae
```

We can check the status via metroae. This will also show us the data directory mount point that was created during install.

```
[caso@metroae-host ~]$ metroae container status

>>> Checking docker version

>>> Getting status of the MetroAE container

>>> Getting the output of 'docker ps'

CONTAINER
574c7314aa7e

>>> Getting the versions in the container itself

Container version: "1.1.2"
MetroÆ version: "v3.3.0"
MetroÆ GUI version:  "0.1"

>>> Getting container configuration from host

MetroAE setup type: Config only
MetroAE container data path: /home/caso/metroae_data

[MetroAE v3.3.0, script 1.0.4, container softlaunch]
```

The templates and the VSD API Specification should be downloaded and available in the path provided during setup.

```
[caso@metroae-host ~]$ cd metroae_data/
[caso@metroae-host metroae_data]$ ls -la
total 36
drwxrwxr-x.  7 caso   caso     167 Jul  9 18:25 .
drwx------.  3 caso   caso      82 Jul  9 18:25 ..
-rw-r--r--.  1 root   root       0 Jul  9 18:25
ansible.log
drwxr-xr-x.  3 root   root      21 Jul  9 18:25
deployments
drwxr-xr-x.  2 root   root    4096 Jul  9 18:25
Documentation
drwxr-xr-x. 15 root   root    4096 Jul  9 18:25 examples
-rw-r--r--.  1 root   root     399 Jul  9 18:25 id_rsa.pub
-rw-r--r--.  1 root   root    5103 Jul  9 18:25 menu
drwxr-xr-x.  5 root   root      72 Jul  9 18:25 standard-
```

```
templates
drwxrwxr-x.  2 centos centos 12288 Jul  9 18:25 vsd-api-
specifications
```

7. Create and source an RC file.

   As detailed in the Environment Variables guide we can use a RC file to fulfill the requirements of the command line. The format required is per below, replacing the values to fit the specific Setup

```
[caso@metroae-host metroae_data]$ cat metroaerc
export TEMPLATE_PATH=/metroae_data/standard-
templates/templates
export USER_DATA_PATH=/metroae_data/config/
export VSD_SPECIFICATIONS_PATH=/metroae_data/vsd-api-
specifications
export VSD_URL=https://20.100.1.3:8443
export VSD_USERNAME=csproot
export VSD_PASSWORD=csproot
export VSD_ENTERPRISE=csp
```

   Source the new file.

```
[caso@metroae-host metroae_data]$ source metroaerc
```

   Can check the user environment

```
[caso@metroae-host metroae_data]$ export | grep -E
"VSD|_PATH"
declare -x TEMPLATE_PATH="/metroae_data/standard-
templates/templates"
declare -x USER_DATA_PATH="/metroae_data/config/"
declare -x VSD_ENTERPRISE="csp"
declare -x VSD_PASSWORD="csproot"
declare -x VSD_SPECIFICATIONS_PATH="/metroae_data/vsd-api-
specifications"
declare -x VSD_URL="https://20.100.1.3:8443"
declare -x VSD_USERNAME="csproot"
```

metroae config is now ready to use. You can get more information about usage by referring to MetroAE Config Usage

MetroAE Config - Usage

General Command Structure

You can use MetroAE Config to configure your VSD via CLI using a local command (`metroae`) that runs on the host where the MetroAE container is installed. All MetroAE Config commands are structured with the menu `metroae config`. For example:

```
metroae config <action> <input>
```

Entering `metroae config`, `metroae config help` or `metroae config --help` will display the valid actions.

```
[root@metroae-host-ebc config]# metroae config

Nuage Networks Metro Automation Engine (MetroAE) Version:
v4.1.0

MetroAE config is a tool that you can use to apply and manage
day-zero configurations
for a Nuage Networks VSD. MetroAE config is only available via
the MetroAE container.
system inside the MetroAE container. To access metroae config
help you can
execute 'metroae config -h'. This will list the positional
arguments that are
supported by the tool. To get additional help for each
positional argument,
execute 'metroae config <positional argument> -h', e.g.
'metroae config create -h'.

MetroAE config usage:

metroae config
Configure VSD
metroae config help
Displays the help text for MetroAE configuration
metroae config version
Displays the current configuration engine version
metroae config engine update
Update configuration engine to the latest version
metroae config <positional args>
Execute configuration engine inside the container
```

```
usage: metroae config [-h]


{create,revert,validate,update,templates,schema,example,document,ve

                      ...

Version 1.0 - This tool reads JSON or Yaml files of templates
and user-data to
write a configuration to a VSD or to revert (remove) said
configuration.

positional arguments:

{create,revert,validate,update,templates,schema,example,document,ve


optional arguments:
  -h, --help            show this help message and exit
```

We can see from the above that the `metroae config` command is split between engine management commands and Configuration execution commands that interact with the VSD.

Engine management commands

You use these commands to interact with the container, check the version, and update the engine when needed.

```
metroae config help
Displays the help text for levistate
metroae config version
Displays the current levistate version
metroae config engine update
Update levistate to the latest version
```

Configuration execution commands

You use these commands to prepare, validate and execute configuration on the VSD.

```
metroae config <positional args>
Execute configuration engine inside the container

usage: metroae config [-h]
```

```
{create,revert,validate,update,templates,schema,example,document,ve

                      ...

Version 1.0 - This tool reads JSON or Yaml files of templates
and user-data to
write a configuration to a VSD or to revert (remove) said
configuration.

positional arguments:

{create,revert,validate,update,templates,schema,example,document,ve
```

For each of the execution commands, you can type `metroae config <action> --help` to obtain information on the action's input requirements. The required input varies depending on the action. For example:

```
[root@metroae-host-ebc config]# metroae config create --help
Nuage Networks Metro Automation Engine (MetroAE) Version:
v4.1.0

MetroAE config is a tool that you can use to apply and manage
day-zero configurations
for a Nuage Networks VSD. MetroAE config is only available via
the MetroAE container.
system inside the MetroAE container. To access metroae config
help you can
execute 'metroae config -h'. This will list the positional
arguments that are
supported by the tool. To get additional help for each
positional argument,
execute 'metroae config <positional argument> -h', e.g.
'metroae config create -h'.

MetroAE config usage:


usage: metroae config create [-h] [-tp TEMPLATE_PATH] [--
version]
                             [-sv SOFTWARE_VERSION] [-sp
SPEC_PATH]
                             [-dp DATA_PATH] [-d DATA] [-v
VSD_URL]
                             [-u USERNAME] [-p PASSWORD] [-c
CERTIFICATE]
                             [-ck CERTIFICATE_KEY] [-e
ENTERPRISE] [--debug]
                             [-lf LOG_FILE] [-ll LOG_LEVEL]
```

```
                            [datafiles [datafiles ...]]

positional arguments:
  datafiles             Optional datafile

optional arguments:
  -h, --help            show this help message and exit
  -tp TEMPLATE_PATH, --template-path TEMPLATE_PATH
                        Path containing template files. Can
also set using
                        environment variable TEMPLATE_PATH
  --version             Displays version information
  -sv SOFTWARE_VERSION, --software-version SOFTWARE_VERSION
                        Override software version for VSD. Can
also set using
                        environment variable SOFTWARE_VERSION
  -sp SPEC_PATH, --spec-path SPEC_PATH
                        Path containing object specifications.
Can also set
                        using environment variable
VSD_SPECIFICATIONS_PATH
  -dp DATA_PATH, --data-path DATA_PATH
                        Path containing user data. Can also set
using
                        environment variable USER_DATA_PATH
  -d DATA, --data DATA  Specify user data as key=value
  -v VSD_URL, --vsd-url VSD_URL
                        URL to VSD REST API. Can also set using
environment
                        variable VSD_URL
  -u USERNAME, --username USERNAME
                        Username for VSD. Can also set using
environment
                        variable VSD_USERNAME
  -p PASSWORD, --password PASSWORD
                        Password for VSD. Can also set using
environment
                        variable VSD_PASSWORD
  -c CERTIFICATE, --certificate CERTIFICATE
                        Certificate used to authenticate with
VSD. Can also
                        set using environment variable
VSD_CERTIFICATE
  -ck CERTIFICATE_KEY, --certificate-key CERTIFICATE_KEY
                        Certificate Key used to authenticate
with VSD. Can
                        also set using environment variable
VSD_CERTIFICATE
  -e ENTERPRISE, --enterprise ENTERPRISE
                        Enterprise for VSD. Can also set using
environment
                        variable VSD_ENTERPRISE
  --debug               Output in debug mode
  -lf LOG_FILE, --log-file LOG_FILE
                        Write logs to specified file. Can also
set using
                        environment variable LOG_FILE
```

```
  -ll LOG_LEVEL, --log-level LOG_LEVEL
                        Specify log level (OUTPUT, ERROR, INFO,
DEBUG, API)
                        Can also set using environment variable
LOG_LEVEL
```

Authentication parameters

When interacting with VSD it is required to provide the authentication parameters for the VSD, this includes:

- VSD URL (-v or --vsd-url)
- VSD Enterprise (-e or --enterprise)
- VSD Username (-u or --username)
- VSD Password (-p or --password)

It is also required that the engine is provided location of the data required to execute. This includes:

- User data location (-dp or --data-path)
- Feature Template location (-tp or --template-path)
- VSD API Specifications location (-sp or --spec-path)

These attributes can be provided as an environment RC file. You can get more information about using the environment variables with an RC file by referring to Using an RC Environment File.

Config Actions

The supported actions are generally grouped to those that interact with VSD, maintain and explore supported feature templates, and validate the metroae version and local host settings.

Service Configuration Summary

- create - Creates a new VSD configuration based on the provided user data.

- revert - Reverts a previously created configuration based on the provided user data
- validate - Locally validates that the user data provided meets the feature template requirements
- update - Update configuration in VSD based on provided user data. The update feature is limited in the Features that can be updated and has strict data requirements for its usage. Its provided in this release as a sample of whats possible only.

Template Summary

- templates list - List all supported feature templates
- templates update - Pull latest feature templates from repository
- schema - Dump the feature template schema
- example - Dump the required user data for a feature template.
- document - render usage document of a feature template

Service Configuration Action Details

In all examples unless otherwise specified an RC file is being used and sourced to provide input attributes required for the specified action.

create

Description

Use create to push a configuration that is based on the provided user data to a VSD. When creating a configuration you must provide the configuration engine with the user data file for configuration, or if a directory is specified then all user data templates in that directory will be used.

Usage

The create command is executed by providing a single user data template or a directory.

metroae config create - creates all user-data templates in the default USER_DATA_PATH directory
metroae config create <user-data.yml> - creates the user-data template specified
metroae config create <user-data.yml> --data-path /path-to/<new-directory> - creates the user data template specified that is located in the directory /path-to/<new-directory>

Example

This example creates an enterprise using the user data "enterprise.yml". It calls a single template “Enterprise” and inputs minimal user data.

```
- template: Enterprise
  values:
    enterprise_name: SimpleEnterprise
```

When using the create action as previously outlined you need to provide VSD location and credentials. In the below case they are sourced to metroae command via user environment variables. To create the VSD Enterprise from our user data template list above we use the create action and pass in the name of the user data template ie. `admin-enterprise-default.yaml.`

```
[root@metroae-host-ebc metroae_data]# metroae config create
admin-enterprise-default.yml

Device: Nuage Networks VSD 6.0.5
    EnterpriseProfile
        forwardingClass = [{'forwardingClass': 'A',
'loadBalancing': False}, {'forwardingClass': 'B',
'loadBalancing': False}, {'forwardingClass': 'C',
'loadBalancing': False}, {'forwardingClass': 'D',
'loadBalancing': False}, {'forwardingClass': 'E',
'loadBalancing': False}, {'forwardingClass': 'F',
'loadBalancing': False}, {'forwardingClass': 'G',
'loadBalancing': False}, {'forwardingClass': 'H',
'loadBalancing': False}]
        VNFManagementEnabled = False
```

```
        description = 'profile for SimpleEnterprise'
        enableApplicationPerformanceManagement = False
        name = 'profile_SimpleEnterprise'
        [store id to name enterprise_profile_id]
    Enterprise
        enterpriseProfileID = [retrieve enterprise_profile_id
(EnterpriseProfile:id)]
        name = 'SimpleEnterprise'
        description = 'enterprise SimpleEnterprise'
>>> All actions successfully applied
```

The output of the creation action lists attribute level creation, including the default values that will be used where the user-data template did not provide any values. The attributes listed in the output are the attribute names as they appear in VSD API. The enterprise will now be created in VSD.

It is important to note here that the config engine is determining what version of VSD is being configured, from time to time attributes across major versions may change in the API and the engine uses the VSD version to determine what feature template to use. For example if we create the same Enterprise in a 5.4.1 VSD (note that I am overriding the local environment variable and parsing in a different VSD URL in the command line).

```
[root@metroae-host-ebc config]# metroae config create admin-
enterprise-default.yml --vsd-url https://20.100.1.103:8443

Device: Nuage Networks VSD 5.4.1
    EnterpriseProfile
        VNFManagementEnabled = False
        allowedForwardingClasses = ['A', 'B', 'C', 'D', 'E',
'F', 'G', 'H']
        description = 'profile for SimpleEnterprise'
        enableApplicationPerformanceManagement = False
        name = 'profile_SimpleEnterprise'
        [store id to name enterprise_profile_id]
    Enterprise
        enterpriseProfileID = [retrieve enterprise_profile_id
(EnterpriseProfile:id)]
        name = 'SimpleEnterprise'
        description = 'enterprise SimpleEnterprise'
>>> All actions successfully applied
```

We can see that there are differences between the API attributes for Forwarding Classes.

In 5.4.1 it is specified as

```
        allowedForwardingClasses = ['A', 'B', 'C', 'D', 'E',
'F', 'G', 'H']
```

In 6.0.5 it is specified as

```
        forwardingClass = [{'forwardingClass': 'A',
'loadBalancing': False}, {'forwardingClass': 'B',
'loadBalancing': False}, {'forwardingClass': 'C',
'loadBalancing': False}, {'forwardingClass': 'D',
'loadBalancing': False}, {'forwardingClass': 'E',
'loadBalancing': False}, {'forwardingClass': 'F',
'loadBalancing': False}, {'forwardingClass': 'G',
'loadBalancing': False}, {'forwardingClass': 'H',
'loadBalancing': False}]
```

The engine determined based on the VSD version and the feature template what attribute is required, while the user data used to create the Enterprise remains consistent.

revert

Description

The revert action deletes configuration from the VSD. It must not be mistaken for a catch all delete action as we must have all the user-data present in order to revert the configuration. The same user-data template that was used in the create action will be parsed and used when reverting.

Attempting to revert user data that is not present in VSD results in a silent pass.

Usage

The create command is executed by providing a single user data template or a directory.

metroae config revert - validates all user data templates in the default USER_DATA_PATH directory
metroae config revert <user-data.yml> - validates the user data template specified
metraoe config revert <user-data.yml> --data-path /path-to/<new-directory> - validates the user data template specified that is located in the directory /path-to/<new-directory>

Example

This example uses the same user-data from enterprise.yml in the create action but with deleting the DemoEnterprise we created in VSD in the previous example.

```
[root@metroae-host-ebc config]# metroae config revert admin-
enterprise-default.yml

Device: Nuage Networks VSD 6.0.5
Revert Enterprise (template: Enterprise)
Revert EnterpriseProfile (template: Enterprise)
>>> All actions successfully applied
```

The output is less verbose as the create, the revert outputs template level actions. The DemoEnterprise will now be deleted in VSD. Note that if the DemoEnterprise had other objects configured within it during the create and revert actions above, and those object block a deletion then the revert would fail.

validate

Description

The validate action allows you to check the userdata against the feature template requirements and provides you with a list of other default attributes that will be configured.

Usage

The validate command is executed by providing a single user data template or a directory.

metroae config validate - validates all user data templates in the default USER_DATA_PATH directory
metroae config validate <user-data.yml> - validates the user data template specified
metroae config validate <user-data.yml> --data-path /path-to/<new-directory> - validates the user data template specified that is located in the directory /path-to/<new-directory>

As the validate command only runs locally any VSD specific parameters are ignored.

Example

This example uses the `admin-enterprise-default.yaml` user data template from the previous create and revert examples.

With the validate command you can check that the user data in the template is sufficient for the requirements (this of course doesn't validate any conflicts in pre-existing VSD configuration) along with detailing the default values that will be configured that were not specified in the user-data.

```
[root@metroae-host-ebc config]# metroae config validate admin-
enterprise-default.yml

Device: Nuage Networks VSD 6.0.5
Configuration
    EnterpriseProfile
        forwardingClass = [{'forwardingClass': 'A',
'loadBalancing': False}, {'forwardingClass': 'B',
'loadBalancing': False}, {'forwardingClass': 'C',
'loadBalancing': False}, {'forwardingClass': 'D',
'loadBalancing': False}, {'forwardingClass': 'E',
'loadBalancing': False}, {'forwardingClass': 'F',
'loadBalancing': False}, {'forwardingClass': 'G',
'loadBalancing': False}, {'forwardingClass': 'H',
'loadBalancing': False}]
        VNFManagementEnabled = False
        description = 'profile for SimpleEnterprise'
        enableApplicationPerformanceManagement = False
        name = 'profile_SimpleEnterprise'
        [store id to name enterprise_profile_id]
    Enterprise
        enterpriseProfileID = [retrieve enterprise_profile_id
(EnterpriseProfile:id)]
        name = 'SimpleEnterprise'
        description = 'enterprise SimpleEnterprise'
```

```
>>> All actions valid
```

In the above case the RC file specified a VSD that is currently running 6.0.5 and thus validation occurred against a version of the Enterprise template that supports 6.0.5. If we want to validate one time against a VSD that is running a different version, say for example our 5.4.1 VSD from earlier besides editing the environment variable we can:

1. Override the VSD URL from the RC file in the command line, here I know that the VSD at the specified URL is running 5.4.1

```
[root@metroae-host-ebc config]# metroae config validate admin-
enterprise-default.yml --vsd-url https://20.100.1.103:8443

Device: Nuage Networks VSD 5.4.1
Configuration
    EnterpriseProfile
        VNFManagementEnabled = False
        allowedForwardingClasses = ['A', 'B', 'C', 'D', 'E',
'F', 'G', 'H']
        description = 'profile for SimpleEnterprise'
        enableApplicationPerformanceManagement = False
        name = 'profile_SimpleEnterprise'
        [store id to name enterprise_profile_id]
    Enterprise
        enterpriseProfileID = [retrieve enterprise_profile_id
(EnterpriseProfile:id)]
        name = 'SimpleEnterprise'
        description = 'enterprise SimpleEnterprise'

>>> All actions valid
```

1. Specify the version to be used in the command line directly

```
[root@metroae-host-ebc config]# metroae config validate admin-
enterprise-default.yml --software-version 5.4.1

Configuration
    EnterpriseProfile
        VNFManagementEnabled = False
        allowedForwardingClasses = ['A', 'B', 'C', 'D', 'E',
'F', 'G', 'H']
        description = 'profile for SimpleEnterprise'
        enableApplicationPerformanceManagement = False
        name = 'profile_SimpleEnterprise'
```

```
        [store id to name enterprise_profile_id]
    Enterprise
        enterpriseProfileID = [retrieve enterprise_profile_id
(EnterpriseProfile:id)]
        name = 'SimpleEnterprise'
        description = 'enterprise SimpleEnterprise'

>>> All actions valid
```

In the later case we can see that the engine did not determine the version of the VSD and thus the version output is silent, it used the input from the CLI. As shown above in the create example we can again see differences from the first validation using a 6.0.5 VSD vs the later using 5.4.1 we can see a key difference in the attributes that will be pushed to VSD.

5.4.1

```
        allowedForwardingClasses = ['A', 'B', 'C', 'D', 'E',
'F', 'G', 'H']
```

6.0.5

```
        forwardingClass = [{'forwardingClass': 'A',
'loadBalancing': False}, {'forwardingClass': 'B',
'loadBalancing': False}, {'forwardingClass': 'C',
'loadBalancing': False}, {'forwardingClass': 'D',
'loadBalancing': False}, {'forwardingClass': 'E',
'loadBalancing': False}, {'forwardingClass': 'F',
'loadBalancing': False}, {'forwardingClass': 'G',
'loadBalancing': False}, {'forwardingClass': 'H',
'loadBalancing': False}]
```

Template Commands Detail

templates list

Description

The templates list command provides a list of the standard feature templates that are available on the local host in the

current template-path directory. The feature-templates are returned in Alphabetical order.

Usage

The templates list command requires only the location of the feature templates.

metroae config templates list - provides a list of feature templates available in the default TEMPLATE_PATH directory
metroae config templates list --template-path /path-to/<template-directory> - provides a list of feature templates available in the directory /path-to/<template-directory"

Example

In this example we are reading in the supported feature templates from the default directory specified in the RC file.

```
[root@metroae-host-ebc config]# metroae config templates list

Application
Application Binding
Application Performance Management
Application Performance Management Binding
Bgp Neighbor
Bidirectional Security Policy
Bidirectional Security Policy Entry
Bridge Port
Captive Portal Profile
Cos Remarking Policy
DC Gateway
DC Gateway Port
DHCP Option
DHCP Pool
DHCPv6 Option
Destination Url
Dscp Remarking Policy
Egress Qos Policy
Enterprise
Enterprise Permission
Gateway Vlan
Infrastructure Access Profile
Infrastructure Gateway Profile
Infrastructure Vsc Profile
Ingress Qos Policy
L2 Domain
L3 Domain
```

```
Monitorscope
NSG Access Port
NSG Network Port
NSG Patch Profile
NSG ZFBInfo Download
NSGateway
NSGateway Activate
NSGateway Template
Network Performance Binding
Network Performance Measurement
Performance Monitor
Policy Group
Rate Limiter
Routing Policy
SSID Connection
Service Chaining Policy
Static Route
Subnet
Symmetric Qos Policy
Underlay
VSD User
Wifi Port
ZFB Auto Assignment
Zone
```

templates update

Description

The feature templates are frequently augmented. As new product features are released or new solutions developed and documented the list of supported and standard templates will not be static. A user must be able to update these templates on demand. To support this the template update command is provided to download the latest set of standard feature templates.

The feature templates and VSD API Specification will be installed in the data directory provided as the mount point for the container.

Usage

metroae config templates update - download the latest templates and VSD API Specification

Example

```
[root@metroae-host-ebc metroae_data]# ls -la
<snip>
drwxr-xr-x.  5 root   root        72 May 13 15:29 standard-
templates
drwxrwxr-x.  2 centos centos   12288 May 13 15:29 vsd-api-
specifications
[root@oc-ebc-config-1 metroae]# metroae config templates update
Updating templates...

[root@metroae-host-ebc metroae_data]# ls -la
<snip>
drwxr-xr-x.  5 root   root        72 May 13 18:45 standard-
templates
drwxrwxr-x.  2 centos centos   12288 May 13 18:45 vsd-api-
specifications
```

schema

Description

Each supported feature template includes its own schema. The schema defines the required user data and format for the feature. The template schema command provides the ability to dump the feature template schema details without the user having to dig through files manually.

Usage

The template schema command requires the input of a single template. If there are a VSD version specific templates due to VSD API changes then we can optionality add the VSD version.

metroae config schema <template-name> - Display the available user data input and requirements for the specified template.

```
[root@metroae-host-ebc metroae_data]# metroae config schema --
help
Nuage Networks Metro Automation Engine (MetroAE) Version:
v4.1.0

MetroAE config is a tool that you can use to apply and manage
day-zero configurations
```

```
for a Nuage Networks VSD. MetroAE config is only available via
the MetroAE container.
system inside the MetroAE container. To access metroae config
help you can
execute 'metroae config -h'. This will list the positional
arguments that are
supported by the tool. To get additional help for each
positional argument,
execute 'metroae config <positional argument> -h', e.g.
'metroae config create -h'.

MetroAE config usage:


usage: metroae config schema [-h] [-tp TEMPLATE_PATH] [--
version]
                             [-sv SOFTWARE_VERSION]
                             [template_names [template_names
...]]

positional arguments:
  template_names        Template names

optional arguments:
  -h, --help            show this help message and exit
  -tp TEMPLATE_PATH, --template-path TEMPLATE_PATH
                        Path containing template files. Can
also set using
                        environment variable TEMPLATE_PATH
  --version             Displays version information
  -sv SOFTWARE_VERSION, --software-version SOFTWARE_VERSION
                        Override software version for VSD. Can
also set using
                        environment variable SOFTWARE_VERSION
```

Example

In the previous examples of Configuration actions we used the Enterprise feature template to create and revert an enterprise in VSD. The user data template that was required to support this is detailed within those examples. However we can use the “template schema” command to detail all the configuration options available to us.

```
[root@metroae-host-ebc metroae_data]# metroae config schema
Enterprise

{
  "title": "Schema validator for Nuage Metro Config template
Enterprise",
```

```
  "$id": "urn:nuage-metro:config:template:enterprise",
  "required": [
    "enterprise_name"
  ],
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "enable_application_performance_management": {
      "type": "boolean",
      "description": "optional enablement of the AAR feature
suite. Defaults to disabled.",
      "title": "Enable application performance management"
    },
    "allow_gateway_management": {
      "type": "boolean",
      "description": "optional enablement of gateway management
within the Enterprise (not csproot). Defaults to disabled.",
      "title": "Allow gateway management"
    },
    "vnf_management_enabled": {
      "type": "boolean",
      "description": "optional enablement VNF hosting on VNS
NSGs. Defaults to disabled.",
      "title": "Vnf management enabled"
    },
    "description": {
      "type": "string",
      "description": "optional description of the enterprise.
Defaults to \"enterprise <enterprise_name>\".",
      "title": "Description"
    },
    "enterprise_name": {
      "type": "string",
      "description": "name of the Enterprise being created.",
      "title": "Enterprise name"
    },
    "routing_protocols_enabled": {
      "type": "boolean",
      "description": "optional enablement of Routing protocols
in the Enterprise. Defaults to disabled.",
      "title": "Routing protocols enabled"
    },
    "allow_advanced_qos_configuration": {
      "type": "boolean",
      "description": "optional enablement of Advanced QoS
features. Defaults to disabled.",
      "title": "Allow advanced qos configuration"
    },
    "forwarding_classes": {
      "items": {
        "enum": [
          "A",
          "B",
          "C",
          "D",
          "E",
          "F",
```

```
          "G",
          "H"
        ]
      },
      "type": "array",
      "description": "optional list of enabled forwarding
classes. Defaults to all classes.",
      "title": "Forwarding classes"
    },
    "local_as": {
      "type": "integer",
      "title": "Local as"
    },
    "enterprise_profile_name": {
      "type": "string",
      "description": "optional for attaching an existing
Organization Profile to the new Enterprise.",
      "title": "Enterprise profile name"
    },
    "dhcp_lease_interval": {
      "type": "integer",
      "description": "optional lease time which is returned in
DHCP Offers. Defaults to 24.",
      "title": "Dhcp lease interval"
    },
    "floating_ips_quota": {
      "type": "integer",
      "description": "optional number of floating IPs that can
be assigned within the Enterprise. Defaults to 0.",
      "title": "Floating ips quota"
    },
    "encryption_management_mode": {
      "enum": [
        "disabled",
        "managed"
      ],
      "description": "optional enablement of Encryption
features within the Enterprise. Defaults to disabled.",
      "title": "Encryption management mode"
    },
    "load_balancing_classes": {
      "items": {
        "enum": [
          "A",
          "B",
          "C",
          "D",
          "E",
          "F",
          "G",
          "H"
        ]
      },
      "type": "array",
      "description": "optional list of load balancing
classes.",
      "title": "Load balancing classes"
```

```
    },
    "allow_trusted_forwarding_classes": {
      "type": "boolean",
      "description": "optional enablement of DSCP trust.
Defaults to disabled.",
      "title": "Allow trusted forwarding classes"
    }
  }
}
```

Again to check the schema against a specific VSD API version add --software-version

example

Description

Each feature template requires a set of user data to execute a configuration action, the actual user data and format is specifi to the feature template in question. The “template example” command provides a formatted sample of the user data required for a specific feature template. It provides the configuration options, value types and will let you know whether the value is optional (if not optional, then it is mandatory).

Usage

The template example command requires the input of a single template. Again if there are VSD version dependencies the software version can be added.

metroae config example <template-name> - Display a sample user-data template for the specified template.

```
[root@metroae-host-ebc metroae_data]# metroae config example --
help
Nuage Networks Metro Automation Engine (MetroAE) Version:
v4.1.0

MetroAE config is a tool that you can use to apply and manage
day-zero configurations
for a Nuage Networks VSD. MetroAE config is only available via
the MetroAE container.
system inside the MetroAE container. To access metroae config
```

```
help you can
execute 'metroae config -h'. This will list the positional
arguments that are
supported by the tool. To get additional help for each
positional argument,
execute 'metroae config <positional argument> -h', e.g.
'metroae config create -h'.

MetroAE config usage:


usage: metroae config example [-h] [-tp TEMPLATE_PATH] [--
version]
                              [-sv SOFTWARE_VERSION]
                              [template_names [template_names
...]]

positional arguments:
  template_names        Template names

optional arguments:
  -h, --help            show this help message and exit
  -tp TEMPLATE_PATH, --template-path TEMPLATE_PATH
                        Path containing template files. Can
also set using
                        environment variable TEMPLATE_PATH
  --version             Displays version information
  -sv SOFTWARE_VERSION, --software-version SOFTWARE_VERSION
                        Override software version for VSD. Can
also set using
                        environment variable SOFTWARE_VERSION
```

Example

This example uses the Enterprise template.

```
[root@metroae-host-ebc metroae_data]# metroae config example
Enterprise

# Create "tenant" in VSD with the Enterprise feature template.
An Enterprise is sometimes referred to as an Organization, or a
Partition in the OpenStack use case.
- template: Enterprise
  values:
    - enterprise_name: ""                       # (string) name
of the Enterprise being created.
      description: ""                           # (opt string)
optional description of the enterprise. Defaults to "enterprise
<enterprise_name>".
      enterprise_profile_name: ""               # (opt
reference) optional for attaching an existing Organization
Profile to the new Enterprise.
      forwarding_classes: []                    # (opt list of
```

```
choice) optional list of enabled forwarding classes. Defaults
to all classes.
      load_balancing_classes: []               # (opt list of
choice) optional list of load balancing classes.
      routing_protocols_enabled: False         # (opt boolean)
optional enablement of Routing protocols in the Enterprise.
Defaults to disabled.
      local_as: 0                              # (opt integer)
      dhcp_lease_interval: 0                   # (opt integer)
optional lease time which is returned in DHCP Offers. Defaults
to 24.
      vnf_management_enabled: False            # (opt boolean)
optional enablement VNF hosting on VNS NSGs. Defaults to
disabled.
      allow_advanced_qos_configuration: False  # (opt boolean)
optional enablement of Advanced QoS features. Defaults to
disabled.
      allow_gateway_management: False          # (opt boolean)
optional enablement of gateway management within the Enterprise
(not csproot). Defaults to disabled.
      allow_trusted_forwarding_classes: False  # (opt boolean)
optional enablement of DSCP trust. Defaults to disabled.
      enable_application_performance_management: False # (opt
boolean) optional enablement of the AAR feature suite. Defaults
to disabled.
      encryption_management_mode: disabled     # (opt
['disabled', 'managed']) optional enablement of Encryption
features within the Enterprise. Defaults to disabled.
      floating_ips_quota: 0                    # (opt integer)
optional number of floating IPs that can be assigned within the
Enterprise. Defaults to 0.
```

Per earlier examples, to check against a specific VSD API version add --software-version.

```
[root@metroae-host-ebc metroae_data]# metroae config example
Enterprise --software_version 5.4.1

# Create "tenant" in VSD with the Enterprise feature template.
An Enterprise is sometimes referred to as an Organization, or a
Partition in the OpenStack use case.
- template: Enterprise
  values:
    - enterprise_name: ""                      # (string) name
of the Enterprise being created.
      description: ""                          # (opt string)
optional description of the enterprise. Defaults to "enterprise
<enterprise_name>".
      enterprise_profile_name: ""              # (opt
reference) optional for attaching an existing Organization
Profile to the new Enterprise.
      forwarding_classes: []                   # (opt list of
choice) optional list of enabled forwarding classes. Defaults
```

```
to all classes.
      routing_protocols_enabled: False         # (opt boolean)
optional enablement of Routing protocols in the Enterprise.
Defaults to disabled.
      local_as: 0                              # (opt integer)
      dhcp_lease_interval: 0                   # (opt integer)
optional lease time which is returned in DHCP Offers. Defaults
to 24.
      vnf_management_enabled: False            # (opt boolean)
optional enablement VNF hosting on VNS NSGs. Defaults to
disabled.
      allow_advanced_qos_configuration: False  # (opt boolean)
optional enablement of Advanced QoS features. Defaults to
disabled.
      allow_gateway_management: False          # (opt boolean)
optional enablement of gateway management within the Enterprise
(not csproot). Defaults to disabled.
      allow_trusted_forwarding_classes: False  # (opt boolean)
optional enablement of DSCP trust. Defaults to disabled.
      enable_application_performance_management: False # (opt
boolean) optional enablement of the AAR feature suite. Defaults
to disabled.
      encryption_management_mode: disabled     # (opt
['disabled', 'managed']) optional enablement of Encryption
features within the Enterprise. Defaults to disabled.
      floating_ips_quota: 0                    # (opt integer)
optional number of floating IPs that can be assigned within the
Enterprise. Defaults to 0.
```

document

Description

Feature templates are provided with embedded documentation that describes the template, provides usage context, parameter/attribute descriptions, requirements and format along with any restrictions. The document command provides a direct way of extracting the documentation for a template. Documentation is provided in md format.

Note that at this point in time template specific documentation is only provided locally when you install and setup metroae config and run the document command.

Usage

The document command can be executed on a single template, in which case the rendered document is printed to the terminal, or can be executed against a directory. either via specifying the `template-path` in an RC file or overriding on the command line.

metroae config document <template> - outputs documentation for the feature template specified
metroae config document - creates md files for all templates in the directory specified in the local environment variables
metroae config document --template-path /path-to/<templates-dir> - creates md files for all templates in the directory specified in the local environment variables

Example

To generate the documentation for the Enterprise template

```
[root@metroae-host-ebc metroae_data]# metroae config document
Enterprise

## Feature Template: Enterprise
#### Description
Create "tenant" in VSD with the Enterprise feature template. An
Enterprise is sometimes referred to as an Organization, or a
Partition in the OpenStack use case.

#### Usage
Each Enterprise in VSD is defined with an attached Organization
Profile which enables additional feature sets in an Enterprise.
Typically you would create an Organization Profile, create an
Enterprise, then attach the profile to the Enterprise.

The Enterprise feature template automatically creates an
Organization Profile for each new Enterprise based on provided
user data. Thus, defining Enterprise and enabling advanced
features for that Enterprise are handled in one *create* step.

You can create an Enterprise with an existing Organization
Profile simply by specifying which Organization Profile
(enterprise_profile_name) to use.

#### Template File Name
/metroae_data/standard-templates/templates/enterprise_v600.yml

#### User Data Requirements
If you do not provide values for the optional parameters listed
below, then default values are used.
```

```
# Create "tenant" in VSD with the Enterprise feature template.
An Enterprise is sometimes referred to as an Organization, or a
Partition in the OpenStack use case.
- template: Enterprise
  values:
    - enterprise_name: ""                      # (string) name
of the Enterprise being created.
      description: ""                          # (opt string)
optional description of the enterprise. Defaults to "enterprise
<enterprise_name>".
      enterprise_profile_name: ""              # (opt
reference) optional for attaching an existing Organization
Profile to the new Enterprise.
      forwarding_classes: []                   # (opt list of
choice) optional list of enabled forwarding classes. Defaults
to all classes.
      load_balancing_classes: []               # (opt list of
choice) optional list of load balancing classes.
      routing_protocols_enabled: False         # (opt boolean)
optional enablement of Routing protocols in the Enterprise.
Defaults to disabled.
      local_as: 0                              # (opt integer)
      dhcp_lease_interval: 0                   # (opt integer)
optional lease time which is returned in DHCP Offers. Defaults
to 24.
      vnf_management_enabled: False            # (opt boolean)
optional enablement VNF hosting on VNS NSGs. Defaults to
disabled.
      allow_advanced_qos_configuration: False  # (opt boolean)
optional enablement of Advanced QoS features. Defaults to
disabled.
      allow_gateway_management: False          # (opt boolean)
optional enablement of gateway management within the Enterprise
(not csproot). Defaults to disabled.
      allow_trusted_forwarding_classes: False  # (opt boolean)
optional enablement of DSCP trust. Defaults to disabled.
      enable_application_performance_management: False # (opt
boolean) optional enablement of the AAR feature suite. Defaults
to disabled.
      encryption_management_mode: disabled     # (opt
['disabled', 'managed']) optional enablement of Encryption
features within the Enterprise. Defaults to disabled.
      floating_ips_quota: 0                    # (opt integer)
optional number of floating IPs that can be assigned within the
Enterprise. Defaults to 0.



#### Parameters
Name | Required | Type | Description
---- | -------- | ---- | -----------
enterprise_name | required | string | name of the Enterprise
being created.
description | optional | string | optional description of the
enterprise. Defaults to "enterprise <enterprise_name>".
enterprise_profile_name | optional | reference | optional for
attaching an existing Organization Profile to the new
```

```
Enterprise.
forwarding_classes | optional | list | optional list of enabled
forwarding classes. Defaults to all classes.
load_balancing_classes | optional | list | optional list of
load balancing classes.
routing_protocols_enabled | optional | boolean | optional
enablement of Routing protocols in the Enterprise. Defaults to
disabled.
local_as | optional | integer |
dhcp_lease_interval | optional | integer | optional lease time
which is returned in DHCP Offers. Defaults to 24.
vnf_management_enabled | optional | boolean | optional
enablement VNF hosting on VNS NSGs. Defaults to disabled.
allow_advanced_qos_configuration | optional | boolean |
optional enablement of Advanced QoS features. Defaults to
disabled.
allow_gateway_management | optional | boolean | optional
enablement of gateway management within the Enterprise (not
csproot). Defaults to disabled.
allow_trusted_forwarding_classes | optional | boolean |
optional enablement of DSCP trust. Defaults to disabled.
enable_application_performance_management | optional | boolean
| optional enablement of the AAR feature suite. Defaults to
disabled.
encryption_management_mode | optional | choice | optional
enablement of Encryption features within the Enterprise.
Defaults to disabled.
floating_ips_quota | optional | integer | optional number of
floating IPs that can be assigned within the Enterprise.
Defaults to 0.


#### Restrictions
**create:**
* You must include a value for enterprise_name in the template.
* The enterprise_name must be unique to the VSD.
* If Routing Protocols are enabled, then a local_as must be
included.
* Feature enablement must have entitlement license in place.

**revert:**
* You cannot revert an Enterprise which has domains.
* You cannot revert an Enterprise which has activated NSGs.
```

We can see that the above was for a VSD 6.0.0 version. If we wanted documentation for a 5.4.1 version we can add the `--software-version 5.4.1` to the command.

To generate md files for all templates we can run the document command with no additional output, it will process all templates provided in the template-path env variable.

```
[root@metroae-host-ebc metroae_data]# metroae config document

Generating documentation
Writing Application documentation to documentation/template-
application.md
Writing Application Binding documentation to
documentation/template-applicationbinding.md
Writing Application Performance Management documentation to
documentation/template-applicationperformancemanagement.md
Writing Application Performance Management Binding
documentation to documentation/template-
applicationperformancemanagementbinding.md
Writing Bgp Neighbor documentation to documentation/template-
bgp-neighbor.md
Writing Bidirectional Security Policy documentation to
documentation/template-bd-sec-policy.md
Writing Bidirectional Security Policy Entry documentation to
documentation/template-bd-sec-policy-entry.md
Writing Bridge Port documentation to documentation/template-
bridge-port.md
Writing Captive Portal Profile documentation to
documentation/template-nsg-wifi-captive-portal-profile.md
Writing Cos Remarking Policy documentation to
documentation/template-cos-remarking-policy.md
Writing DC Gateway documentation to documentation/template-dc-
gateway.md
Writing DC Gateway Port documentation to
documentation/template-dcgateway-port.md
Writing DHCP Option documentation to documentation/template-
dhcp-option.md
Writing DHCP Pool documentation to documentation/template-dhcp-
pool.md
Writing DHCPv6 Option documentation to documentation/template-
dhcpv6option.md
Writing Destination Url documentation to
documentation/template-destination-url.md
Writing Dscp Remarking Policy documentation to
documentation/template-dscp-remarking-policy.md
Writing Egress Qos Policy documentation to
documentation/template-egress-qos-policy.md
Writing Enterprise documentation to documentation/template-
enterprise.md
Writing Enterprise Permission documentation to
documentation/template-enterprise-permission.md
Writing Gateway Vlan documentation to documentation/template-
gateway-vlan.md
Writing Infrastructure Access Profile documentation to
documentation/template-infrastructure-access-profile.md
Writing Infrastructure Gateway Profile documentation to
documentation/template-infrastructure-gateway-profile.md
Writing Infrastructure Vsc Profile documentation to
documentation/template-infrastructure-vsc-profile.md
Writing Ingress Qos Policy documentation to
documentation/template-ingress-qos-policy.md
Writing L2 Domain documentation to documentation/template-
```

```
l2domain.md
Writing L3 Domain documentation to documentation/template-
l3domain.md
Writing Monitorscope documentation to documentation/template-
nsg-monitor-scope.md
Writing NSG Access Port documentation to
documentation/template-nsg-access-port.md
Writing NSG Network Port documentation to
documentation/template-nsg-network-port.md
Writing NSG Patch Profile documentation to
documentation/nsgpatchprofile.md
Writing NSG ZFBInfo Download documentation to
documentation/template-nsg-iso-download.md
Writing NSGateway documentation to documentation/template-nsg-
instance.md
Writing NSGateway Activate documentation to
documentation/template-nsg-activate.md
Writing NSGateway Template documentation to
documentation/template-nsg-template.md
Writing Network Performance Binding documentation to
documentation/template-nsg-network-performance-measurement-
binding.md
Writing Network Performance Measurement documentation to
documentation/template-nsg-network-performance-measurement.md
Writing Performance Monitor documentation to
documentation/template-nsg-performance-monitor.md
Writing Policy Group documentation to documentation/template-
policy-group.md
Writing Rate Limiter documentation to documentation/template-
rate-limiter.md
Writing Routing Policy documentation to documentation/template-
routing-policy.md
Writing SSID Connection documentation to
documentation/template-nsg-wifi-ssid.md
Writing Service Chaining Policy documentation to
documentation/template-service-chaining-policy.md
Writing Static Route documentation to documentation/template-
static-route.md
Writing Subnet documentation to documentation/template-
subnet.md
Writing Symmetric Qos Policy documentation to
documentation/template-symmetric-qos.md
Writing Underlay documentation to documentation/template-
underlay.md
Writing VSD User documentation to documentation/template-vsd-
user.md
Writing Wifi Port documentation to documentation/template-nsg-
wifi-port.md
Writing ZFB Auto Assignment documentation to
documentation/template-nsg-zfb.md
Writing Zone documentation to documentation/template-zone.md
```

Reduce Command Requirements with an RC File

The configuration engine supports the use of local environment variable in the user shell. Assuming the VSD, templates and VSP API SPEC are not changing locally this reduces the command requirements to the action and a single data input. See the article Exporting the environment variables using an RC File for details.

Without using an RC file

```
metroae config create admin-enterprise-default.yaml --template-
path /metroae_data/standard-templates/matroae-
templates/templates --spec-path /metroae_data/vsd-api-
specifications/vsd-api-specifications --data-path
/metroae_data/config/ --vsd-url https://20.100.1.103:8443 --
username csproot --password csproot --enterprise csp
```

With the use of an RC file

```
metroae create admin-enterprise-default.yaml
```

MetroAE Config - Simplified Operations by Exporting Environment Variables With an RC File

In order to execute actions against the VSD, you must provide local path information and authentication credentials to the configuration engine.

Instead of specifying each requirement at the command line individually, you can pass the information to the engine with environment variables by:

1. Creating an RC file on the local client which sets the required variables in your current shell, and
2. sourcing that file.

Create RC File

Create a file containing the following parameters. See parameter details below.

```
export TEMPLATE_PATH=/metroae_data/standard-templates/templates
export USER_DATA_PATH=/metroae_data/configuration/
export VSD_SPECIFICATIONS_PATH=/metroae_data/vsd-api-
specifications
export VSD_URL=https://10.101.0.28:8443
export VSD_USERNAME=csproot
export VSD_PASSWORD=csproot
export VSD_ENTERPRISE=csp
```

Parameter Details

The first three parameters specify paths on the local client where the configuration engine is running.

TEMPLATE_PATH - path to feature templates. Default is shown below. Do not change the path unless you are using non-standard/modified templates.

```
TEMPLATE_PATH=/metroae_data/standard-templates/templates
```

USER_DATA_PATH - location/directory of user data templates that will be used to configure VSD. Sample templates are provided in the default installation. However, in most cases the directory is user-defined.

```
USER_DATA_PATH=/metroae_data/configuration/
```

VSD_SPECIFICATIONS_PATH - location/path of the VSD API Specifications. The specifications are downloaded and installed in the default location shown below as part of template install/update.

```
VSD_SPECIFICATIONS_PATH=/metroae_data/vsd-api-specifications
```

The next four parameters specify details about configuring the target VSD.

VSD_URL - URL of the target VSD to be configured

```
VSD_URL=https://10.101.0.28:8443
```

VSD_USERNAME - username of the account that will be configuring the VSD. This username is used by the administrator to access the VSD UI. Default is shown below.

```
VSD_USERNAME=csproot
```

VSD_PASSWORD - password of the account that will be configuring the VSD. This is the password that is used by the administrator to access the VSD UI. Default is shown below.

```
VSD_PASSWORD=csproot
```

VSD_ENTERPRISE - enterprise that the user account is configured to access in the VSD. Default is shown below.

```
VSD_ENTERPRISE=csp
```

Source RC File

You give the configuration engine access to the environment parameters for your user environment by sourcing the file you created with the command below, substituting `<rcfilename>` with the name of your RC file.

```
[root@metroae]# source <rcfilename>
```

Overriding Environment Variables

If, for any environment variable, you want to specify a different value than the one specified in the RC file, you don't need to copy the file and source it again. You can override any environment value at the command line by adding the option *–datapath* to the command.

For example, if your RC file specifies the user data path as:

```
USER_DATA_PATH=/metroae_data/configuration/
```

And you want to create an enterprise with the user data path as /metroae_data/demo/ instead, then run the command below to

override the RC file value.

```
[root@metroae]# metroae config create enterprise-default.yaml -
-datapath /metroae_data/demo/
```

Overriding is supported for all command line variables. You can find out more about command line usage by referring to MetroAE Command Line Usage.

MetroAE Config Feature Template Overview

What is a Feature Template

Feature Templates are what is used to define the data required to create one or many objects in VSD. They are yaml files that include Jinja2 substitution to normalize the user data provided, and to allow for data defaults and abstraction and call specific metroae config functions that operate against the VSD API.

MetroAE Config functions

In order to support the creation and linking of configuration within VSD, MetroAE Config supports a series of key functions. These include - create, select, store and retrieve.

- create - used to define a new object in the VSD
- select - used to find or place the creation of an object within the correct hierarchy
- store and retrieve - used to “lookup” a UUID of an object that will later be used within the create function.

Each feature template calls each of these functions based on the requirements of the feature. For instance if we want to create a new subnet in a L3 Domain we would require the following.

Enterprise --> Domain --> Zone --> Subnet

The subnet template would define the following

```
select enterprise
  select domain
    select zone
      create subnet
```

Interacting with Templates

As covered in the general overview section MetroAE Config provides methods for checking the supported Feature Templates, outputting sample data required for execution of a template, checking the schema and rendering the support documentation for the specific template.

As a user the sample data and documentation will inform you what is required to create VSD configuration based on the template. One of the first things we have to do when building networks with VSD is create an Enterprise. Lets take a specific look then at the Enterprise Feature Template.

We can dump out the possible configuration options of the template by asking for an example.

```
[caso@metroae-host metroae_data]$ metroae config example
"Enterprise"

# Create "tenant" in VSD with the Enterprise feature template.
An Enterprise is sometimes referred to as an Organization, or a
Partition in the OpenStack use case.
- template: Enterprise
  values:
    - enterprise_name: ""                      # (string) name
of the Enterprise being created.
      description: ""                          # (opt string)
optional description of the enterprise. Defaults to "enterprise
<enterprise_name>".
      enterprise_profile_name: ""              # (opt
reference) optional for attaching an existing Organization
Profile to the new Enterprise.
      forwarding_classes: []                   # (opt list of
choice) optional list of enabled forwarding classes. Defaults
to all classes.
      load_balancing_classes: []               # (opt list of
choice) optional list of load balancing classes.
      routing_protocols_enabled: False         # (opt boolean)
optional enablement of Routing protocols in the Enterprise.
Defaults to disabled.
      local_as: 0                              # (opt integer)
      dhcp_lease_interval: 0                   # (opt integer)
optional lease time which is returned in DHCP Offers. Defaults
to 24.
      vnf_management_enabled: False            # (opt boolean)
optional enablement VNF hosting on VNS NSGs. Defaults to
disabled.
      allow_advanced_qos_configuration: False  # (opt boolean)
optional enablement of Advanced QoS features. Defaults to
disabled.
      allow_gateway_management: False          # (opt boolean)
```

```
optional enablement of gateway management within the Enterprise
(not csproot). Defaults to disabled.
      allow_trusted_forwarding_classes: False  # (opt boolean)
optional enablement of DSCP trust. Defaults to disabled.
      enable_application_performance_management: False # (opt
boolean) optional enablement of the AAR feature suite. Defaults
to disabled.
      encryption_management_mode: disabled     # (opt
['disabled', 'managed']) optional enablement of Encryption
features within the Enterprise. Defaults to disabled.
      floating_ips_quota: 0                    # (opt integer)
optional number of floating IPs that can be assigned within the
Enterprise. Defaults to 0.
```

By running example against the feature template name, in this case “Enterprise” we are provided will all the attributes that can be used when creating an Enterprise. We can also see that only a single attribute is mandatory, that being the "enterprise_name". All other attributes are listed as "opt".

We can then define user data based on the example to create an Enterprise in VSD, in this case we will take all the default values and only specify the mandatory attributes.

```
- template: Enterprise
  values:
    enterprise_name: SimpleEnterprise
```

Creating VSD Configuration using a Feature Template

To this point we have decided on the object we need to create in the VSD, ie. an Enterprise, we have used MetroAE Config to tell us what parameters are required to create the object and we have defined a set of data based on that. Now we can create the Enterprise in the VSD.

Its important at this point to highlight when creating configuration with feature templates there is no need to tell MetroAE Config during execution what object its creating. We have already defined that when we created the user data above. Before executing lets take a closer look at what the data is actually specifying.

From above here is the data we are going to use:

```
- template: Enterprise
  values:
    enterprise_name: SimpleEnterprise
```

A simple set of 3 lines, but what is this telling the config engine?

```
- template: Enterprise
```

This first line is critical, this is the key that tells the engine to create this set of data based on the “Enterprise” template. The definition of the Enterprise template specifies the actions that need to be performed in the creation of the new object. Now lets take a look at the data we are specifying, this always belongs to “values:”

```
values:
  enterprise_name: SimpleEnterprise
```

In this case we are only specifying an Enterprise name which is "SimpleEnterprise". If we were to want to specify more options when creating the Enterprise they would also be included as part of the value set. For instance the below would define an Enterprise with many more features enabled.

```
- template: Enterprise
  values:
    enterprise_name: DemoEnterprise
    description: "All the good stuff enabled"
    local_as: 65000
    forwarding_classes: [A, B, C, D]
    bgp_enabled: True
    vnf_management_enabled: True
    allow_advanced_qos_configuration: True
    allow_gateway_management: True
    allow_trusted_forwarding_classes: True
    enable_application_performance_management: True
    encryption_management_mode: managed
    floating_ips_quota: 100
```

Now as the data set provided tells the config engine what template to use, and the template has the operations that are required in order to create the object the execution is then a generic "create". Thus when creating configuration in VSD the data always specifies the objects and the user does not need to know specific runtime syntax for the myriad of features that VSD supports.

So lets execute.

Where are the Feature Templates

Feature templates are downloaded and installed within your metroae_data directory when you setup MetroAE Config. Assuming things were left with defaults this would be "standard-templates/templates/".

Updating Feature Templates

Updating of feature templates is covered in the general config usage section. However it can be executed using "metroae config templates update".

Two items of note when Updating

- The existing standard-template directory will be overwritten when executing updates (see recommendations below on Modifying/Creating templates)
- Internet access will be required. There is nothing special here, we have tried to make the install and setup process as simple as possible. If offline access is necessary the templates can be updated on any internet connected host and copy the “standard-templates” and “vsd-api-specifications” directories to your offline host.

Modifying and Creating Feature Templates

If you want to modify or create a new feature template, we recommend you copy the standard-template directory and

modify within that directory. You can then either create and source another rc file that contains the “new” templates directory or pass in the new directory at run time with "-tp" or “–template-path” with the path of your new or modified templates.

More information on Feature Templates, Samples and how to use data wisely

Template List

Feature templates are expected to be delivered orthogonally to MetroAE releases. The list of templates will be expanding to coverage of more VSD features. The list of templates supported in your current MetroAE setup can be gathered via "metroae config templates list", which provides an alphabetical list of all Feature Templates.

Sample Data for Templates

We have provided sample data for each feature template. This sample yaml files are named with the template category, feature template and any other possible options. In some cases multiple samples for the same template maybe provided.

How to Define user data

The MetroAE Config has some powerful features in terms of being able to reduce data required by the user by functions such as inheritance and substitution. The section on Inheritance will attempt to describe those.

MetroAE Config - Inheritance

When creating configuration user data to pass into MetroAE config you can make use of inheritance to avoid having to specify common parameters multiple times.

User data files are able to specify groups that contain common values for parameters that will then be inherited (used) by subsequent feature configuration.

Using Groups

You create a group by giving it a name and specifying the parameter names and associated values you would like contained in that group. The parameter names and values will be inherited by any feature that references that group.

For example to create a group called StaticRoutes that specifies an Enterprise and L3 Domain to use you would include the following in your user data:

```
- group: StaticRoutes
  values:
    enterprise_name: DemoEnterprise
    domain_name: L3-Domain-US
```

You can then reference this group instead of repeating the same value when instantiating a feature template with your configuration values.

If feature configuration that references a group contains parameters that are also specified in that group then the values in the feature configuration will take precedence over (override) the values defined in the group.

MetroAE config supports two types of inheritance using groups:

- Parent/Child

- Substitution

Parent/Child inheritance

With Parent/Child inheritance you instantiate one or more feature templates as children of the group you have defined. All children inherit all properties (and their values) from their parent group.

For example to create three static routes, two of which are within the same L3 Domain, under the same Enterprise you could include the following in your user data: network-static-route-groups-child2.yaml

```
- group: StaticRoutes
  values:
    enterprise_name: DemoEnterprise
    domain_name: L3-Domain-US
  children:
    - template: Static Route
      values:
        - address: 172.16.0.0
          netmask: 255.255.0.0
          next_hop: 100.1.1.10
        - address: 172.25.0.0
          netmask: 255.255.0.0
          next_hop: 200.1.1.20

- template: Static Route
  values:
  - enterprise_name: DemoEnterprise
    domain_name: L3-Domain-EMEA
    address: 172.17.0.0
    netmask: 255.255.0.0
    next_hop: 100.1.2.15
```

which would generate output similar to:

```
(example)# metroae config create network-static-route-groups-
child2.yaml

Device: Nuage Networks VSD 6.0.1
    [select Enterprise (name of DemoEnterprise)]
        [select Domain (name of L3-Domain-US)]
            StaticRoute
                netmask = '255.255.0.0'
```

```
                nextHopIp = '100.1.1.10'
                address = '172.16.0.0'
            StaticRoute
                netmask = '255.255.0.0'
                nextHopIp = '200.1.1.20'
                address = '172.25.0.0'
        [select Domain (name of L3-Domain-EMEA)]
            StaticRoute
                netmask = '255.255.0.0'
                nextHopIp = '100.1.2.15'
                address = '172.17.0.0'
```

Inheritance by substitution

Inheritance by substitution allows you to insert the parameter values defined in a group into the configuration for a feature without having to specify the resulting configuration as children of the group. This provides more flexibility in how you can structure your user data.

For example to create the same three static routes, two of which are within the same L3 Domain, under the same Enterprise you could include the following in your user data:

network-static-route-groups-sub2.yaml

```
- group: L3-Domain-US
  values:
    enterprise_name: DemoEnterprise
    domain_name: L3-Domain-US

- group: L3-Domain-EMEA
  values:
    enterprise_name: DemoEnterprise
    domain_name: L3-Domain-EMEA

- template: Static Route
  values:
    - $group_domain: L3-Domain-US
      address: 172.16.0.0
      netmask: 255.255.0.0
      next_hop: 100.1.1.10
    - $group_domain: L3-Domain-EMEA
      address: 172.17.0.0
      netmask: 255.255.0.0
      next_hop: 100.1.2.15
    - $group_domain: L3-Domain-US
      address: 172.25.0.0
```

```
      netmask: 255.255.0.0
      next_hop: 200.1.1.20
```

which would generate output similar to:

```
(example)# metroae config create network-static-route-groups-
sub2.yaml

Device: Nuage Networks VSD 6.0.1
    [select Enterprise (name of DemoEnterprise)]
        [select Domain (name of L3-Domain-US)]
            StaticRoute
                netmask = '255.255.0.0'
                nextHopIp = '100.1.1.10'
                address = '172.16.0.0'
            StaticRoute
                netmask = '255.255.0.0'
                nextHopIp = '200.1.1.20'
                address = '172.25.0.0'
        [select Domain (name of L3-Domain-EMEA)]
            StaticRoute
                netmask = '255.255.0.0'
                nextHopIp = '100.1.2.15'
                address = '172.17.0.0'
```

Combining Parent/Child inheritance with substitution

It is possible to use Parent/Child inheritance as well as inheritance by substitution together in a single user data file and even together within the configuration of a single feature.

For example to create the same three static routes, with their netmask only specified once you could include the following in your user data: network-static-route-groups-child-sub.yaml

```
- group: L3-Domain-US
  values:
    enterprise_name: DemoEnterprise
    domain_name: L3-Domain-US

- group: L3-Domain-EMEA
  values:
    enterprise_name: DemoEnterprise
    domain_name: L3-Domain-EMEA
```

```
- group: StaticRoutes
  values:
    $group_domain: L3-Domain-US
    netmask: 255.255.0.0
  children:
    - template: Static Route
      values:
        - address: 172.16.0.0
          next_hop: 100.1.1.10
        - address: 172.25.0.0
          next_hop: 200.1.1.20
        - $group_domain: L3-Domain-EMEA
          address: 172.17.0.0
          next_hop: 100.1.2.15
```

which would generate output similar to:

```
(example)# metroae config create network-static-route-groups-
child-sub.yaml

Device: Nuage Networks VSD 6.0.1
    [select Enterprise (name of DemoEnterprise)]
        [select Domain (name of L3-Domain-US)]
            StaticRoute
                netmask = '255.255.0.0'
                nextHopIp = '100.1.1.10'
                address = '172.16.0.0'
            StaticRoute
                netmask = '255.255.0.0'
                nextHopIp = '200.1.1.20'
                address = '172.25.0.0'
        [select Domain (name of L3-Domain-EMEA)]
            StaticRoute
                netmask = '255.255.0.0'
                nextHopIp = '100.1.2.15'
                address = '172.17.0.0'
```
