SENSE Northbound API Program

v1.2

1. Introduction

The SENSE system provides the mechanisms to enable multi-domain orchestration for a wide variety of network and other cyberinfrastructure resources in a highly customized manner. These orchestrated services can be customized for individual domain science workflow systems and requirements. SENSE services include Layer 2 Point to Point Network Connections, Layer 2 Multipoint Network Topologies, and Layer 3 Virtual Private Network (VPN) services. Additional information regarding the SENSE System is available in the following paper:

- Software-Defined Network for End-to-end Networked Science at the Exascale, Elsevier Future Generation Computer Systems, Volume 110, September 2020, Pages 181-201, https://doi.org/10.1016/j.future.2020.04.018, (Accepted Manuscript: https://arxiv.org/abs/2004.05953)

The SENSE system includes the following key components:

- SENSE Orchestrator - This is the point where the user interacts with the SENSE system. This interaction can be via a Graphical User Interface (GUI) or a programmatic Application Programming Interface (API). The Orchestrator is also responsible for interacting the underlying network and end systems resources via the Resource Managers as described below.

- SENSE Resource Manager - This is a distributed set of resources which interact with a variety of network and end-system resources to construct the end-to-end networked services.

This document describes how to use the SENSE Northbound API.

This includes the following steps:

- Account Request
- Account Configuration for API
- Northbound API Use (Services Instantiation Workflow)

These steps are described in more detail below.

2. SENSE Orchestrator User Account Establishment

The following steps are defined to establish a SENSE Orchestrator User Account.

2.1) Account Request

Request an account via Orchestrator portal:

- https://sense-o-east.es.net:8443/StackV-web/

- Use CILogon Option to select appropriate identity provider and login
- This will result in a review and approval process by the SENSE administrators.

2.2) Account Configuration for API

- The user should then login to the Orchestrator using CILogon option
- The user should create a username/password based logon via:
    - Click on "Account" at the top right
    - Select "Password" and create a Password for this account

After this, the user will be able to login to the SENSE Portal using their Username/Password or using CILogin. The Username will be the email used when initially requesting portal access via CIlogon. The Password can be changed via the same process noted above.

The Username/Password credentials will be used for Northbound API access.

3) Install Northbound API Client Library

A SENSE Northbound API client library has been uploaded to pipi.org. The following steps describe how to install this library.

- Python3 Install
  verify Python3 is installed on the local machine. If not, install Python3.

- Client Library Install
  pip3 install sense-o-api==1.24

  For latest version check: https://pypi.org/project/sense-o-api/

4) Northbound API Use (Services Instantiation Workflow)

There are multiple services, methods, and workflows that can be instantiated using this client library. Additional detailed description of these options, and more technical details, are available at the below locations:

- SENSE Northbound API Client - Swagger Definition
    - https://app.swaggerhub.com/apis/xi-yang/SENSE-O-Intent-API/2.0.2#

- SENSE Northbound API Client - Software Repository
    - https://github.com/sdn-sense/sense-o-py-client
    - Package distribution, install, use
        - https://github.com/sdn-sense/sense-o-py-client/wiki/Package-distribution,-install-and-use

The steps below show simple use case for the following workflow and service scenario:

- User runs Northbound API from their machine under a local user account
- The Services request is for a Layer 2 Point-to-Point path
- The two endpoints for this path are known
- The user will instantiate and delete the Layer 2 Point-to-Point path by manually running as script which calls the SENSE Orchestrator Client Library
- The only thing the user will adjust between instantiations is the bandwidth requested.

a) Configuration File Creation
Create a client library configuration file and add it one of the following locations:

- /etc/sense-o-auth.yaml (library will look here first)
- ~/.sense-o-auth.yaml (library will look here second)

Here is an example sense-o-auth.yaml configuration file.

```
AUTH_ENDPOINT:
https://sense-o-east.es.net:8543/auth/realms/StackV/protocol/openid-connect/token
#sense-o-east
API_ENDPOINT: https://sense-o-east.es.net:8443/StackV-web/restapi
CLIENT_ID: StackV
USERNAME: <sense orchestrator portal account username>
PASSWORD: <sense orchestrator portal account password>
SECRET: <api secret provided by SENSE administrators)
verify: False
#for servers with verifiable SSL cert
#verify: True
```

b) Service Profile
For this type of workflow, the SENSE Administrators will create a "Service Profile" which has the following parameters:

The following Layer 2 Point to Point Service features are defined in advance and fixed:
- Source Port
- Source VLAN
- Destination Port
- Destination VLAN
- Range of Allowable Bandwidth for each instantiation

The following Layer 2 Point to Point Service features can be adjusted by the user in realtime as part of their service instantiation:

- Layer 2 Point to Point Service Bandwidth (within the range defined above)

c) Run Workflow

- There is an example workflow script included as part of the client library install as noted below:
    - sense_util.py

- Run help command to see run options:

  sense_util.py -h

  usage: sense_util.py [-h] [-cr | -ca | -r | -d | -D | -s | -p] [-f FILE] [-u UUID] [-n NAME]

  optional arguments:

-h, --help	show this help message and exit
-cr, --create	create service instance (requires one of optional -f, optional -u)
-ca, --cancel	cancel (and not delete) an existing service instance (requires -u)
-r, --reprovision	reprovision an existing service instance (requires -u)
-d, --delete	cancel and delete service instance (requires -u)
-D, --delete-only	delete service instance (requires -u)
-s, --status	get service instance status (requires -u)
-p, --profile	describe a service profile (requires -u)
-f FILE, --file FILE	service intent request file
-u UUID, --uuid UUID	service profile uuid or instance uuid
-n NAME, --name NAME	service instance alias name

- Review Service Profile
  Use this command to review the Service Profile created above.

    - sense_util.py -p -u <*profile-ID*>

- Instantiate a Service

  Use this command for the simple workflow described above:

    - sense_util.py -cr -f sense-provision.json -n <NAME>

  This requires a service description file with the filename "sense-provision.json"

  An example of this file for provisioning a path using the workflow concept described above is as follows:

  sense-provision.json
  {"service_profile_uuid": "<profile-ID>",

"queries": [{"ask": "edit","options": [{"data.connections[0].bandwidth.capacity": "<bw>"}]}]}

The user will need to edit the following fields based on the specific service they want to instantiate:

- <profile-ID> - The SENSE Orchestrator administrators will set up the "Service Profile" and provide the service_profile-ID to the user. After this, the user can instantiate and delete the service independently.
- <bw> - The user can edit this value as needed for each instantiation. This parameter is specified in units of Mbit/s, and will need to be within the predefined range defined in the "Service Profile".

This command will return a service-ID, which will be needed to check status and delete the service.

- Check Service Status
  Use this command to delete the service created above.
  - sense_util.py -s -u <service-ID>

- Delete the Service
  Use this command to delete the service created above.
  - sense_util.py -d -u <service-ID>

    Note: when creating and deleting services, best practices are to leave at least 2 minutes between completion of the -d (delete) option and a new -cr (create) option command. This provides the network control plane and dataplane sufficient time to complete all changes.

- SENSE Orchestrator Portal Use
  This SENSE Orchestrator portal will also allow the following actions:
  - Instantiate the Service
  - Check Service Status
  - Review visualization of the Service
  - Delete the Service

Appendix A Miscellaneous Notes

B.1) HTTPS Connection Certificate Configuration
The client library configuration file will be located in one of the following locations:

- /etc/sense-o-auth.yaml (library will look here first)
- ~/.sense-o-auth.yaml (library will look here second)

Here is an example sense-o-auth.yaml configuration file.

AUTH_ENDPOINT:
https://sense-o-east.es.net:8543/auth/realms/StackV/protocol/openid-connect/token
#sense-o-east
API_ENDPOINT: https://sense-o-east.es.net:8443/StackV-web/restapi
CLIENT_ID: StackV
USERNAME: <sense orchestrator portal account username>
PASSWORD: <sense orchestrator portal account password>
SECRET: <api secret provided by SENSE administrators)
verify: False
#for servers with verifiable SSL cert
#verify: True

For the "verify" the option, the following should be noted:

- verify: False will result in warning which includes this:

"InsecureRequestWarning: Unverified HTTPS request is being made to host 'sense-o-east.es.net'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.html#ssl-warnings...."

This is because the standard Python3 distribution does not include the latest Incommon Cert distribution. This warning will not impact the client API operation. To prevent this warning from happening, the latest Incommon Certs can be added to the Python3 distribution as follows:

a) Show all the certs from the server:
$ openssl s_client -connect sense-o-east.es.net:8543 -showcerts

b) copy all the Certs, except for Cert 0.
That would be 1, 2 and 3 in this case

b) Find the Python CA bundle file

$ python -c 'import certifi; print(certifi.where())'

Example output:
/usr/local/lib/python3.9/site-packages/certifi/cacert.pem

c) Paste the certs (1,2,3) from above to append to the cacert.pem file

d) After this, set the <verify> parameter to True in the sense-o-auth.yaml config file: verify: True