**SNIA DOCUMENTATION**

**Installation Steps:**

Steps to run emulator in Linux flavors

1🡪Installing python (update python version)

**$ sudo apt-get update**

**$ sudo apt-get –y update**

**Python3-v** (To know Python Version)

2🡪 Installing PIP and Virtual Environment

* To manage packages in python we have to install PIP

**$ sudo apt-get install –y python3-pip**

* Virtualenv enables multiple installations simultaneously, for different projects.

**$ sudo apt-get install python-virtualenv**

3🡪Install FLASK Micro Framework

* Below command will get flask activated in our virtual environment

**$ pip install Flask**

4🡪After completing Python and flask installations we have to install some dependencies for our Project.

Below are some of the dependencies for up and running python, flask and other packages.

**python install.py**

If we can’t install through above install.py we should go through installdependencies.txt and install simultaneously one by one.

**Running Snia.sh :**

We should go to SNIA folder from terminal and run following commands.

**$ chmod u+x snia.sh**

**$ ./Snia.sh**

**Understanding Emulator:**

The emulator.py reads the configuration files and parameters. The code handles the path and controlled by resource\_manager.py. The resource manager loads the dynamic resources that we created for storage services. We are fetching the required attributes from json data.

**Constants.py:**

Declares the path of our project and display different resources with their attributes.

**Storage\_Services.py**

We have created class files for both api and collections. Loads static resources from../redfish/v1/… directory. It performs the operations for type definitions describe in mockups.

**Resource\_manager.py**

Loads the mockups for storage services and adds the api and collection classes defined in storage\_services.py. Loads and constructs the resources that are emulated .Loads both the static and dynamic resources.

**Overview:**

redfish/v1/ will return Service Root entry and redfish/v1/odata will return Service Container instance which odata represents. Elements that are utilized for redfish/swordfish management are..

**Systems:** It’s a resource collection that refers Computer Systems for general application server. In each Computer Systems we will have an entry value of Application Server”. In this Computer Systems we can have both Storage System Collections and System Collections.

**Chassis:** In Chasses resource represents physical containers, Subsystems which operates outside of a system's data are linked either directly or indirectly through this resource.

**Storage Services:** A reference to a ComputerSystemCollection with members of type Computer System that support storage services. These Computer System resources represent systems that support Swordfish storage management services.

**Storage Systems:** A reference to a Storage Service Collection with members that are of type Storage Service. A resource collection that references a set of Storage Service resources. Each Storage Service resource represents the resources and behaviors supported by that storage service.