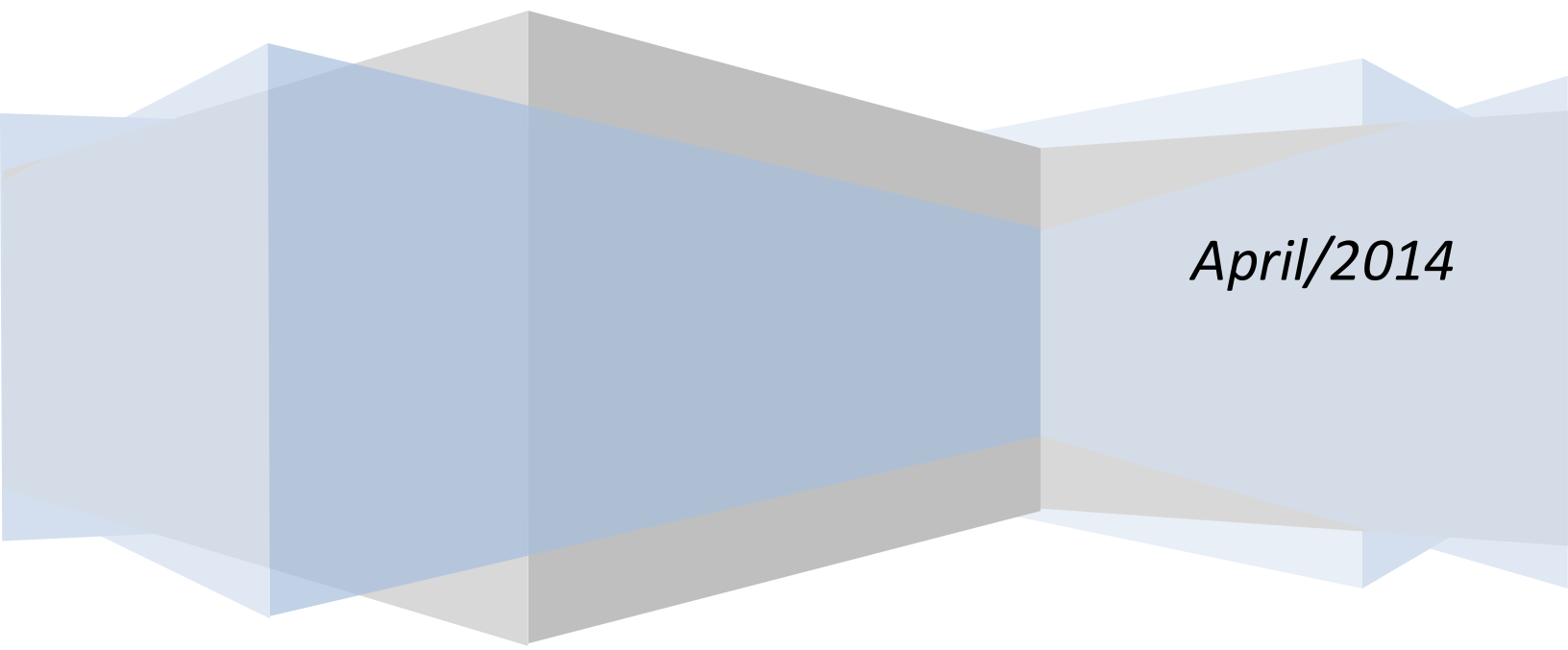


OmniSwitch OpenStack Networking Plug-in Release Notes

Release 1.2 for OpenStack Havana –
OONP_H_R01_2

April/2014

An abstract graphic at the bottom of the page consisting of several overlapping, semi-transparent 3D cubes and rectangular prisms in shades of light blue and grey, creating a modern, architectural look.

Introduction

This readme file provides a quick overview of the Alcatel-Lucent Enterprise (ALUE) OmniSwitch OpenStack Networking Plug-in (OONP) release 1.2 R01 for OpenStack Havana. The plug-in provides OpenStack Networking (Neutron) with the ability to manage the Layer-2 configuration of OmniSwitch devices. The plug-in does not manage the compute host network stack; this is typically managed by the OpenVSwitch (OVS) plug-in and the neutron_openvswitch_agent.

The following ALUE products are supported:

- OS10K
- OS6900
- OS6850E
- OS6855

Contents

Introduction	2
Features	2
Installation notes	3
Configuration options	4
Supported configurations	6
Limitations	9
Additional configuration notes	9

Features

The plug-in supports the following features:

- 802.1q VLAN based tenant networks.
- Multiple physical topologies – ranging from a single switch to multi-switch based core-edge or spine-leaf topologies.
- Edge port automatic configuration based on vNP classification using either: VM MAC address; or 802.1q VLAN tag.
- Automated inter-switch (Uplink Port) VLAN configuration using either static VPA or MVRP.

The following product matrix shows which features are supported/used for physical OmniSwitch configuration:

Feature Mode/Product	OS10K	OS6900	OS6850E/ OS6855
Switch access method	ReST/Telnet	ReST/ Telnet	Telnet

Edge Port Configuration	MAC/VLAN	MAC/VLAN	MAC/VLAN
Uplink Port Configuration	VPA/MVRP	VPA/MVRP	VPA/MVRP

While a mix of switch features is supported by the plug-in, only one feature mode can be used for the OpenStack configuration. I.e. if the ReST method is selected for communication with the switches, that method will be used for communications with ALL of the switches specified in the configuration (which implies that ALL switches must support the ReST method)

Installation notes

The plug-in is installed on the OpenStack controller node and can be performed either: by using the `omniplugin_install` installation script; or by manually extracting the contents of the tar.gz file into a subdirectory named `omniswitch` within the neutron installation plug-in directory (typically `/usr/lib/python2.7/dist-packages/neutron/plugins`)

The `omniplugin_install` script will install the plug-in files into the default OpenStack neutron distribution directory, `/usr/lib/python2.7/dist-packages/neutron/plugins`. The installation script can be given an optional neutron installation directory that overrides the default of `/usr/lib/python2.7/dist-packages/neutron`. The installation script must be executed as root and run in the same directory as the OONP release package. The package will be copied into the `omniswitch` plug-in directory and then extracted, leaving the original distribution package un-changed. A sample configuration file will be copied into the neutron configuration file directory, `/etc/neutron/plugins/omniswitch`. Note: any existing `omniswitch_network_plugin.ini` file will be overwritten.

If you manually extract and install the plug-in, you need to copy the `omniswitch_network_plugin.ini` configuration file from the plug-in directory into the neutron configuration file location. For the Ubuntu Cloud archive distribution this will be `/etc/neutron/plugins/omniswitch`. You must ensure the extracted files and the OmniSwitch plug-in directory (and the configuration file) are owned by the neutron user and group (`chown -R neutron:neutron`).

After a successful installation (either manual or via the script), the following files will be found in the OmniSwitch plug-in directory:

```
-rw-r--r-- 1 neutron neutron 55412 Apr  4 09:38 consumer.py
-rw-r--r-- 1 neutron neutron 0      Mar 28 09:43 __init__.py
-rwxr-xr-x 1 neutron neutron 1543   Apr  4 09:33 omnidb_setup
-rwxr-xr-x 1 neutron neutron 3618   Apr  4 09:33 omniplugin_install
-rwxr-xr-x 1 neutron neutron 979    Apr  4 09:33 omniswitch_clear_db
-rw-r--r-- 1 neutron neutron 1828   Apr  4 09:33 omniswitch_constants.py
-rw-r--r-- 1 neutron neutron 12355   Apr  4 09:34 omniswitch_db_v2.py
-rw-r--r-- 1 neutron neutron 25192   Apr 15 17:27 omniswitch_device_plugin.py
-rw-r--r-- 1 neutron neutron 7260    Apr  9 17:15 omniswitch_network_plugin.ini
-rw-r--r-- 1 neutron neutron 20887   Apr  4 09:34 omniswitch_network_plugin.py
-rw-r--r-- 1 neutron neutron 2602    Apr  4 09:34 omniswitch_neutron_dbutils.py
-rw-r--r-- 1 neutron neutron 22221   Apr  9 11:43 omniswitch_restful_driver.py
-rwxr-xr-x 1 neutron neutron 927     Apr  4 09:35 omniswitch_setup
-rw-r--r-- 1 neutron neutron 12527   Apr 15 17:25 omniswitch_telnet_driver.py
```

```
-rwxr-xr-x 1 neutron neutron 17197 Apr 15 15:33 omniswitch_topology_utils.py
-rwxr-xr-x 1 neutron neutron 806 Apr 4 09:35 restart
-rw-r--r-- 1 neutron neutron 12372 Apr 14 12:19 telnet_driver.py
```

Neutron must be configured to use the OONP at this point. The typical installation will use a combination of the OONP along with the OVS plug-in. The plug-in configuration is defined in two (2) files in the Ubuntu environment on the controller node:

- 1) **/etc/default/neutron-server** – the configuration file is specified using the following entry:
`NEUTRON_PLUGIN_CONFIG="/etc/neutron/plugins/omniswitch/omniswitch_network_plugin.ini"`
- 2) **/etc/neutron/neutron.conf** – the plug-in configuration is specified using the following entries:
`core_plugin = neutron.plugins.omniswitch.omniswitch_network_plugin.OmniSwitchNetworkPluginV2`

Some basic OmniSwitch configuration must be performed prior to using the plug-in. This includes configuring the per port UNPs, MVRP, and classification modes. This configuration is performed by the `omniswitch_setup` script located in **/usr/lib/python2.7/dist-packages/neutron/plugins/omniswitch**. The script uses the topology defined in the `omniswitch_network_plugin.ini` file to set the initial programming of the switches; therefore, you must edit this file prior to running the setup script. (See the configuration options section for details.)

The `omniswitch_setup` script is a wrapper that calls the python script `omniswitch_topology_utils.py` with the `setup_initial` option and the configuration file as arguments. The `omniswitch_topology_utils.py` script can also be used to reset the switch configuration, by using the `clear_initial` option. The following command will ‘undo’ the initial configuration performed by the `omniswitch_setup` script:

```
omniswitch_topology_utils.py clear_initial \
    /etc/neutron/plugins/omniswitch/omniswitch_network_plugin.ini
```

Configuration options

Configuration options may generally be grouped into two (2) classes: plug-in configurations; and device definitions. Plug-in configuration elements define overall OpenStack integration and database configuration details. Device definitions describe the physical configuration details. Within the device class there are two (2) types of configuration elements: general operational modes (i.e. use telnet to communicate with the physical device); and topology definitions, which describe switch and host interconnection details.

The plug-in configuration elements are located in the [PLUGIN] section and include:

`omni_plugin` – specifies the device plug-in to use and **MUST** be set to `neutron.plugins.omniswitch.omniswitch_device_plugin.OmniSwitchDevicePluginV2`.

`ovs_plugin` – specifies the optional use of the OVS plug-in to manage the compute and network node tenant network interface VLAN configurations. In most configurations this will be set to `neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2`.

`connection` – specifies the shared neutron configuration database location. This should be the same configuration that is specified by the connection parameter in the `/etc/neutron/neutron.conf` file.

`reconnect_interval` – specifies how often the plug-in will try to reconnect to the database after being disconnected. Leave the set to the default of 2.

`network_vlan_ranges` – Specifies which network interface and what VLANs will be used for the tenants. The format is `<identifier>:<starting VLAN>:<ending VLAN>`, i.e `network_vlan_ranges = physnet1:2000:2009`. This is the same format that is used in the OVS plug-in configuration file and the identifier is the name that is used in the bridge_mappings element (of the OVS plug-in configuration)

The device configuration elements are located in the [DEVICE] section. The general operational modes are global in nature, that is they apply to ALL switch(es); they are defined by the following elements:

`host_classification` – specifies how the host switch-port 802.1q tag will be configured. 2 options are available: VLAN, which uses the incoming 802.1q tag to automatically create the VLAN association; or MAC_ADDRESS, which uses the VM MAC address to create the VLAN association.

`core_network_config` – specifies what method to use during the creation of inter-switch 802.1q trunks. 2 methods are identified: MVRP, use MVRP to automatically create 802.1q trunks; VPA, the plug-in will automatically configure each switch-port as needed. MVRP is the preferred method unless the topology includes devices that do not support MVRP, in which case you must use the VPA method. This element is not used in single switch configurations.

`switch_access_method` – specifies how the plug-in will communicate with the omniswitch devices. Possible values are: TELNET and REST. This specifies the method used for all the switches defined in the configuration.

The topology elements define switch details that include IP address, product type, login details, and port utilization. The basic format of each includes: IP address, Device type, Login User, Login password, Prompt, Port data. The edge and core elements allow multiple device definitions separated by commas “,”; for example:

```
omni_edge_devices = <device1>, <device2>, ...
```

You may include line breaks after the commas to allow for readability. The device definition format is:

```
<IPAddr>:<Type>:<user>:<password>:<prompt>:<port> [<port>]...[:<port> [<port>]...]
```

Product type identifiers are: OS6900; OS10K; OS6850E; OS6855; OS6450. The login and command prompt parameters are optional and allow the use of non-default switch values.

There are 3 types of topology elements:

`omni_edge_devices` - used to define one, or more, edge switches (where compute and/or network nodes are connected). There are 2 sets of port parameters used by this element: host connections, compute or network nodes; uplink connections, connected to other (core) switches. At

least one edge switch MUST be defined. Multiple ports may be defined by separating them with spaces. An example device entry (using the default login and prompt data, 2 host connections, and 1 uplink connection) is

```
omni_edge_devices = 10.255.99.211:OS6850E: : : :1/1 1/2:1/10
```

`omni_core_devices` – used to define core switches. If core switches are not present in the topology, this element must either: not be present; or commented out. A core switch does not support compute or network node connections; thus, it contains only one set of port parameters, the ports connected to other switches (either additional core switches or terminating edge switches). An example device entry (using the default login and prompt data) is

```
omni_core_devices = 192.168.222.34:OS6900: : : :1/19 1/20
```

`dhcp_server_interface` – defines the connection point for the shared tenant dhcp server. In a typical OpenStack configuration the network node provides tenant network dhcp services via the `neutron_dhcp_agent` and `dnsmasq`; therefore, this entry is usually synonymous with the network node connection point. An example entry is

```
dhcp_server_interface = 192.168.222.33:OS10K: : : :1/18
```

If the optional use of the OVS plug-in is specified (in the `ovs_plugin` element); then the OVS plug-in configuration elements must be included into the OmniSwitch plug-in configuration file (due to limitations in the current OmniSwitch plug-in architecture). All configuration options are grouped in the `[DATABASE]`, `[OVS]`, `[AGENT]`, and `[SECURITYGROUP]` classes.

Supported configurations

The plug-in was developed and tested using the Ubuntu 12.04.4 LTS release and the Canonical Cloud Archives OpenStack Havana release.

The plug-in has been validated in the following physical configurations:

- A) A single OS6900 switch with a combined controller/network node and 1 compute node. The plug-in is configured to use the telnet device driver and vNP based edge configuration using MAC address classification.

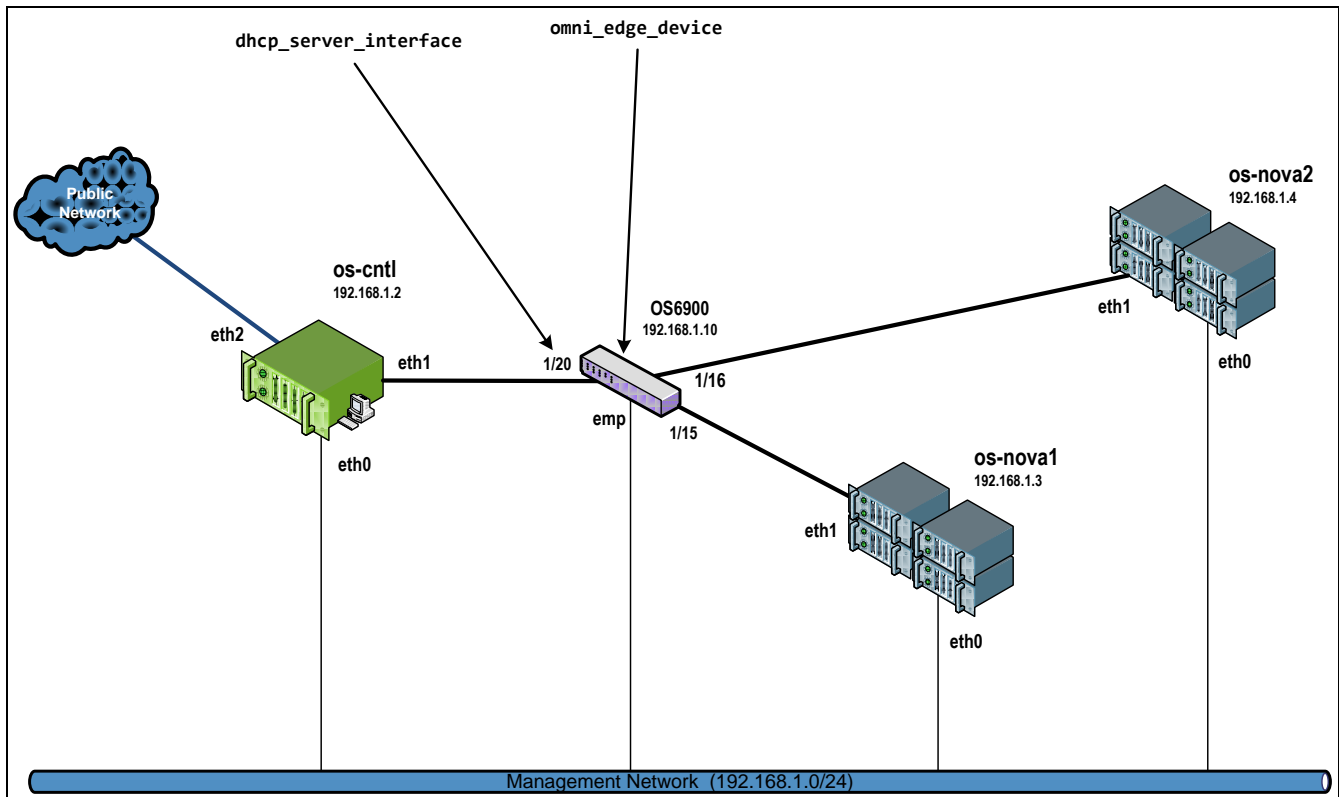


Figure 1. Single Switch Topology

The OmniSwitch plug-in configuration required to realize the topology shown in figure 1 is shown in table 1. The key configuration detail to take note of: is that the **omni_core_devices** configuration element is absent (either missing or commented out is acceptable). The OVS configuration details have been omitted for brevity.

[PLUGIN]

```
omni_plugin = quantum.plugins.omniswitch.omniswitch_device_plugin.OmniSwitchDevicePluginV2
ovs_plugin = quantum.plugins.openvswitch.ovs_quantum_plugin.OVSQuantumPluginV2
sql_connection = mysql://quantum:quantum@192.168.1.2:3306/quantum
reconnect_interval = 2
```

[DEVICE]

```
omni_edge_devices = 192.168.1.10:OS6900: : : :1/15,1/16:
dhcp_server_interface = 192.168.1.10:OS6900: : : :1/20
host_classification = MAC_ADDRESS
core_network_config =
switch_access_method = TELNET
```

- B) A 3 switch core/edge topology using OS6900s, with a combined controller/network node and 2 compute nodes. The plug-in is configured to use the telnet device driver, vNP based edge configuration using MAC address classification, and MVRP to automatically configure the edge switch to core switch

connectivity.

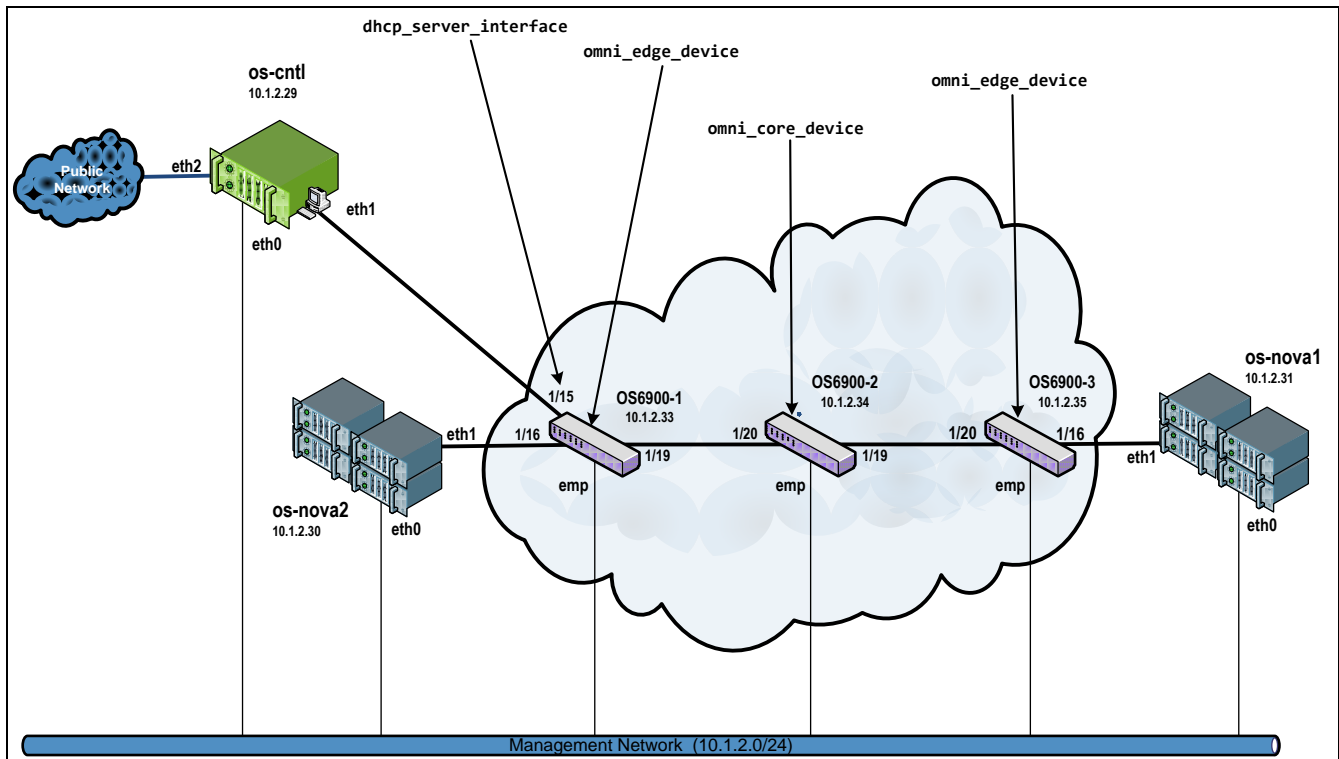


Figure 2. Multi-switch Topology

The OmniSwitch plug-in configuration required to realize the multi-switch topology shown in figure 2 is shown in table 2. Again, the OVS configuration details have been omitted for brevity.

[PLUGIN]

```
omni_plugin = quantum.plugins.omnswitch.omnswitch_device_plugin.OmniSwitchDevicePluginV2
ovs_plugin = quantum.plugins.openvswitch.ovs_quantum_plugin.OVSQuantumPluginV2
sql_connection = mysql://quantum:quantum@10.1.2.29:3306/quantum
reconnect_interval = 2
```

[DEVICE]

```
omni_edge_devices = 10.1.2.33:OS6900: : : 1/16:1/19,
                    10.1.2.35:OS6900: : : 1/16:1/20
omni_core_devices = 10.1.2.34:OS6900: : : 1/19 1/20
dhcp_server_interface = 10.1.2.33:OS6900: : : 1/15
host_classification = MAC_ADDRESS
core_network_config = MVRP
switch_access_method = TELNET
```

Additional supported configurations include the following topologies:

- A) One, or more OS6900s using the ReST device driver.
- B) Multiple OS6900s using either the ReST or telnet device driver and VPA configured connectivity for the edge/core inter-connections.
- C) Multiple OS6900s using either the ReST or telnet device driver, with compute nodes attached to all switches. This is a pure edge-switch topology with no 'core'. In this configuration you will specify only **omni_edge_devices**.

Limitations

The plug-in has not been tested outside of the Ubuntu/Canonical cloud archives environment. The installation scripting may be affected by differences in installation directories and packaging used by other Linux distributions. Additional issues may arise from the python class loader which is partially based on the installation directory layout.

The installation script does NOT check for existing files, it blindly overwrites the OmniSwitch plug-in directory and the OmniSwitch plug-in configuration file. You should save any existing `omniswitch_network_plugin.ini` file prior to (re)running the installation script.

The plug-in does not support the Neutron ML2 device plug-in model.

The OS6860(e) and OS6450 product families are not supported in any configuration.

There is no support for Link-Aggregation (linkagg) configurations in either the switch inter-connections or the host connections.

Shortest-Path-Bridging (SPB) is not supported.

Virtual Chassis configurations are not supported.

Host port configuration based on VPA is not supported.

Floating IP addresses are supported with the following limitations:

- The plug-in will create a phantom tenant network and allocate a VLAN for its use. While the VLAN is configured in the switching infrastructure, it is not active or usable;
- Additionally 2 un-useable IP addresses will be allocated from the floating IP pool; thus, you should take this into account when sizing a floating IP pool.

Additional configuration notes

The neutron-server must be restarted (on the controller node) after making changes to the `omniswitch_network_plugin.ini` file.

The installation script does not set the file ownership or permissions on the plug-in or the configuration files correctly. You must manually set the ownership to the neutron user and neutron group for the plug-in files and the configuration file. Additionally you need to set the permissions on the `omniswitch_topology_utils.py` file to

755 (rwxr-xr-x), found in the OmniSwitch plug-in directory.

The OONP writes log output to the neutron-server log file, which by default is ***/var/log/neutron/server.log***. Output includes plug-in initialization messages and switch configuration commands.

During product validation an improper fcoe configuration element in an OmniSwitch was found to block DHCP DISCOVER messages from VM instances to the network node (running the neutron dhcp-agent). To avoid this either: delete all **fcoe** configuration elements; or ensure that the fcoe priority is set to a non-zero value (a zero value drops all non fcoe traffic)

ALUe has identified a configuration issue related to the OS6850E (and possibly the OS6855), where the VLAN and UNP configurations are not applied to the switch. The workaround is to set the switch CLI prompt to the default (->), and not specify a prompt in that device definition element of the plug-in configuration file.

The use of plug-ins other than the OVS module for host network configuration have not been tested.