Networking Baremetal Documentation

Release 7.3.0.dev18

OpenStack Foundation

Apr 02, 2026

CONTENTS

CHAPTER

ONE

NETWORKING-BAREMETAL PLUGIN

This projects goal is to provide deep integration between the Networking service and the Bare Metal service and advanced networking features like notifications of port status changes and routed networks support in clouds with Bare Metal service.

CHAPTER

TWO

FEATURES

- **L2VNI Mechanism Driver**: Enables baremetal servers to connect to VXLAN and Geneve overlay networks by dynamically allocating VLAN segments and creating OVN localnet ports. See documentation for configuration details.
- **Port Status Notifications**: Real-time notifications of port status changes from the Bare Metal service to the Networking service.
- **Multi-tenant Network Support**: Advanced networking features for baremetal deployments with tenant isolation.
- Free software: Apache license
- Documentation: http://docs.openstack.org/networking-baremetal/latest
- Source: http://opendev.org/openstack/networking-baremetal
- Bugs: https://bugs.launchpad.net/networking-baremetal
- Release notes: https://docs.openstack.org/releasenotes/networking-baremetal/

CHAPTER

THREE

INSTALLATION GUIDE

3.1 Installation

This section describes how to install and configure the `networking-baremetal` plugin and `ironic-neutron-agent`.

The `ironic-neutron-agent` is a neutron agent that populates the host to physical network mapping for baremetal nodes in neutron. Neutron uses this to calculate the segment to host mapping information.

3.1.1 Install the networking-baremetal plugin and agent

At the command line:

```
$ pip install networking-baremetal
```

Or, if you have neutron installed in a virtualenv, install the `networking-baremetal` plugin to the same virtualenv:

```
$ . <path-to-neutron-venv>/bin/activate
$ pip install networking-baremetal
```

Or, use the package from your distribution. For RHEL7/CentOS7:

```
$ yum install python2-networking-baremetal python2-ironic-neutron-agent
```

3.1.2 Enable baremetal mechanism driver in the Networking service

To enable mechanism drivers in the ML2 plug-in, edit the `/etc/neutron/plugins/ml2/ml2_conf.ini` configuration file. For example, this enables the `openvswitch` and `baremetal` mechanism drivers:

```
[ml2]
mechanism_drivers = openvswitch,baremetal
```

3.1.3 Add devices (switches) to manage

The baremetal mechanism ML2 plug-in provides a device driver plug-in interface. If a device driver for the switch model exist the baremetal ML2 plug-in can be configured to manage switch configuration, adding tenant VLANs and setting switch port VLAN configuration etc.

To add a device to manage, edit the `/etc/neutron/plugins/ml2/ml2_conf.ini` configuration file. The example below enables devices: `device_a.example.net` and `device_b.example.net`. Both

devices in the example is using the `netconf-openconfig` device driver. For each device a separate section in configuration defines the device and driver specific configuration.

```
[networking_baremetal]
enabled_devices = device_a.example.net,device_b.example.net

[device_a.example.net]
driver = netconf-openconfig
switch_info = device_a
switch_id = 00:53:00:0a:0a:0a
host = device_a.example.net
username = user
key_filename = /etc/neutron/ssh_keys/device_a_sshkey
hostkey_verify = false

[device_b.example.net]
driver = netconf-openconfig
switch_info = device_b
switch_id = 00:53:00:0b:0b:0b
host = device_a.example.net
username = user
key_filename = /etc/neutron/ssh_keys/device_a_sshkey
hostkey_verify = false
```

3.1.4 Configure ironic-neutron-agent

To configure the baremetal neutron agent, edit the neutron configuration `/etc/neutron/plugins/ml2/ironic_neutron_agent.ini` file. Add an `[ironic]` section. For example:

```
[ironic]
project_domain_name = Default
project_name = service
user_domain_name = Default
password = password
username = ironic
auth_url = http://identity-server.example.com/identity
auth_type = password
os_region = RegionOne
```

3.1.5 Start ironic-neutron-agent service

To start the agent either run it from the command line like in the example below or add it to the init system.

```
$ ironic-neutron-agent \
    --config-dir /etc/neutron \
    --config-file /etc/neutron/plugins/ml2/ironic_neutron_agent.ini \
    --log-file /var/log/neutron/ironic_neutron_agent.log
```

You can create a systemd service file `/etc/systemd/system/ironic-neutron-agent.service` for `ironic-neutron-agent` for systemd based distributions. For example:

```
[Unit]
Description=OpenStack Ironic Neutron Agent
After=syslog.target network.target

[Service]
Type=simple
User=neutron
PermissionsStartOnly=true
TimeoutStartSec=0
Restart=on-failure
ExecStart=/usr/bin/ironic-neutron-agent --config-dir /etc/neutron --config-
↪file /etc/neutron/plugins/ml2/ironic_neutron_agent.ini --log-file /var/log/
↪neutron/ironic-neutron-agent.log
PrivateTmp=true
KillMode=process

[Install]
WantedBy=multi-user.target
```

Note

systemd service file may be already available if you are installing from package released by linux distributions.

Enable and start the `ironic-neutron-agent` service:

```
$ sudo systemctl enable ironic-neutron-agent.service
$ sudo systemctl start ironic-neutron-agent.service
```

CHAPTER

FOUR

CONFIGURATION GUIDE

4.1 Configuration Options

4.1.1 Configuration Reference

The following pages describe configuration options that can be used to adjust the `ironic-neutron-agent` service to your particular situation.

ironic-neutron-agent - Configuration Options

The following is an overview of all available configuration options in networking-baremetal. For a sample configuration file, refer to *Sample Configuration File*.

DEFAULT

debug

Type
boolean

Default
`False`

Mutable
This option can be changed without restarting.

If set to true, the logging level will be set to DEBUG instead of the default INFO level.

log_config_append

Type
string

Default
<None>

Mutable
This option can be changed without restarting.

The name of a logging configuration file. This file is appended to any existing logging configuration files. For details about logging configuration files, see the Python logging module documentation. Note that when logging configuration files are used then all logging configuration is set in the configuration file and other logging configuration options are ignored (for example, log-date-format).

Table 1: Deprecated Variations

Group	Name
DEFAULT	log-config
DEFAULT	log_config

`log_date_format`

Type
string

Default
`%Y-%m-%d %H:%M:%S`

Defines the format string for %(asctime)s in log records. Default: the value above . This option is ignored if log_config_append is set.

`log_file`

Type
string

Default
<None>

(Optional) Name of log file to send logging output to. If no default is set, logging will go to stderr as defined by use_stderr. This option is ignored if log_config_append is set.

Table 2: Deprecated Variations

Group	Name
DEFAULT	logfile

`log_dir`

Type
string

Default
<None>

(Optional) The base directory used for relative log_file paths. This option is ignored if log_config_append is set.

Table 3: Deprecated Variations

Group	Name
DEFAULT	logdir

`use_syslog`

Type
boolean

Default
`False`

Use syslog for logging. Existing syslog format is DEPRECATED and will be changed later to honor RFC5424. This option is ignored if log_config_append is set.

`use_journal`

Type
boolean

Default
`False`

Enable journald for logging. If running in a systemd environment you may wish to enable journal support. Doing so will use the journal native protocol which includes structured metadata in addition to log messages.This option is ignored if log_config_append is set.

`syslog_log_facility`

Type
string

Default
`LOG_USER`

Syslog facility to receive log lines. This option is ignored if log_config_append is set.

`use_json`

Type
boolean

Default
`False`

Use JSON formatting for logging. This option is ignored if log_config_append is set.

`use_stderr`

Type
boolean

Default
`False`

Log output to standard error. This option is ignored if log_config_append is set.

`log_color`

Type
boolean

Default
`False`

(Optional) Set the color key according to log levels. This option takes effect only when logging to stderr or stdout is used. This option is ignored if log_config_append is set.

`log_rotate_interval`

Type
integer

Default
1

The amount of time before the log files are rotated. This option is ignored unless log_rotation_type is set to interval.

`log_rotate_interval_type`

Type
string

Default
days

Valid Values
Seconds, Minutes, Hours, Days, Weekday, Midnight

Rotation interval type. The time of the last file change (or the time when the service was started) is used when scheduling the next rotation.

`max_logfile_count`

Type
integer

Default
30

Maximum number of rotated log files.

`max_logfile_size_mb`

Type
integer

Default
200

Log file maximum size in MB. This option is ignored if log_rotation_type is not set to size.

`log_rotation_type`

Type
string

Default
none

Valid Values
interval, size, none

Log rotation type.

Possible values

interval
Rotate logs at predefined time intervals.

size
Rotate logs once they reach a predefined size.

none
Do not rotate log files.

`logging_context_format_string`

Type
string

Default
```
%(asctime)s.%(msecs)03d %(process)d %(levelname)s %(name)s
[%(global_request_id)s %(request_id)s %(user_identity)s]
%(instance)s%(message)s
```

Format string to use for log messages with context. Used by oslo_log.formatters.ContextFormatter

`logging_default_format_string`

Type
string

Default
```
%(asctime)s.%(msecs)03d %(process)d %(levelname)s %(name)s [-]
%(instance)s%(message)s
```

Format string to use for log messages when context is undefined. Used by oslo_log.formatters.ContextFormatter

`logging_debug_format_suffix`

Type
string

Default
```
%(funcName)s %(pathname)s:%(lineno)d
```

Additional data to append to log message when logging level for the message is DEBUG. Used by oslo_log.formatters.ContextFormatter

`logging_exception_prefix`

Type
string

Default
```
%(asctime)s.%(msecs)03d %(process)d ERROR %(name)s
%(instance)s
```

Prefix each line of exception output with this format. Used by oslo_log.formatters.ContextFormatter

`logging_user_identity_format`

Type
string

Default
```
%(user)s %(project)s %(domain)s %(system_scope)s
%(user_domain)s %(project_domain)s
```

Defines the format string for %(user_identity)s that is used in logging_context_format_string. Used by oslo_log.formatters.ContextFormatter

`default_log_levels`

Type
list

Default
```
['amqp=WARN', 'boto=WARN', 'sqlalchemy=WARN', 'suds=INFO',
'oslo.messaging=INFO', 'oslo_messaging=INFO', 'iso8601=WARN',
'requests.packages.urllib3.connectionpool=WARN', 'urllib3.
connectionpool=WARN', 'websocket=WARN', 'requests.packages.
urllib3.util.retry=WARN', 'urllib3.util.retry=WARN',
'keystonemiddleware=WARN', 'routes.middleware=WARN',
'stevedore=WARN', 'taskflow=WARN', 'keystoneauth=WARN', 'oslo.
cache=INFO', 'oslo_policy=INFO', 'dogpile.core.dogpile=INFO']
```

List of package logging levels in logger=LEVEL pairs. This option is ignored if log_config_append is set.

`publish_errors`

Type
boolean

Default
`False`

Enables or disables publication of error events.

`instance_format`

Type
string

Default
`"[instance: %(uuid)s] "`

The format for an instance that is passed with the log message.

`instance_uuid_format`

Type
string

Default
`"[instance: %(uuid)s] "`

The format for an instance UUID that is passed with the log message.

`rate_limit_interval`

Type
integer

Default
`0`

Interval, number of seconds, of log rate limiting.

`rate_limit_burst`

Type
integer

Default
`0`

Maximum number of logged messages per rate_limit_interval.

`rate_limit_except_level`

Type
string

Default
`CRITICAL`

Valid Values
CRITICAL, ERROR, INFO, WARNING, DEBUG,

Log level name used by rate limiting. Logs with level greater or equal to rate_limit_except_level are not filtered. An empty string means that all levels are filtered.

`fatal_deprecations`

Type
boolean

Default
`False`

Enables or disables fatal status of deprecations.

agent

`report_interval`

Type
floating point

Default
`30`

Seconds between nodes reporting state to server; should be less than agent_down_time, best if it is half or less than agent_down_time.

`log_agent_heartbeats`

Type
boolean

Default
`False`

Log agent heartbeats

baremetal_agent

`enable_ha_chassis_group_alignment`

Type
boolean

Default
`True`

Enable HA chassis group alignment reconciliation for router ports on networks with baremetal external ports. This fixes Launchpad bug #1995078 where mismatched HA chassis group priorities between router gateway ports and baremetal external ports cause intermittent connectivity issues. When enabled, the agent ensures router ports use the same ha_chassis_group as baremetal external ports on the same network.

`ha_chassis_group_alignment_interval`

Type
integer

Default
`600`

Minimum Value
60

Interval in seconds between HA chassis group alignment reconciliation runs. This controls how frequently the agent checks for and fixes mismatched HA chassis groups. Default is 600 seconds (10 minutes). Minimum is 60 seconds to avoid excessive API load.

`limit_ha_chassis_group_alignment_to_recent_changes_only`

Type
boolean

Default
`True`

When enabled, HA chassis group alignment only checks resources created or updated within the time window specified by ha_chassis_group_alignment_window. This reduces reconciliation overhead by focusing on recently created resources that may have mismatched HA chassis groups. When disabled, performs full reconciliation of all resources on each run, which is more thorough but has higher API and database load.

`ha_chassis_group_alignment_window`

Type
integer

Default
`1200`

Minimum Value
0

Time window in seconds for checking recent resources when limit_ha_chassis_group_alignment_to_recent_changes_only is enabled. Default is 1200 seconds (20 minutes), which is 2x the default alignment interval. Resources created or updated within this window will be checked for HA chassis group alignment. Setting to 0 effectively disables windowing even if the limit flag is enabled.

`enable_router_ha_binding`

Type
boolean

Default
`True`

Enable router HA binding for router interface ports on networks with baremetal nodes. When enabled, the agent automatically binds router interface ports to the same HA chassis group as the networks external ports, enabling proper ARP resolution and connectivity between baremetal nodes and their router gateway on VLAN networks. This fixes Launchpad bug #2144458 where baremetal nodes experience persistent connectivity failures to their router gateway. Uses both event-driven binding (for immediate response) and periodic reconciliation (for edge cases).

`enable_router_ha_binding_events`

Type
boolean

Default
`True`

Enable event-driven router HA binding. When enabled, the agent responds immediately to HA chassis group creation events by binding router interface ports on the affected network. This provides instant connectivity when networks are created. Requires enable_router_ha_binding to be enabled. If disabled, only periodic reconciliation will be used, which may result in connectivity delays until the next reconciliation cycle.

`router_ha_binding_interval`

Type
integer

Default
`600`

Minimum Value
60

Interval in seconds for periodic router HA binding reconciliation. This ensures router interface ports are bound to network HA chassis groups even if events are missed or routers are added after the fact. Default is 600 seconds (10 minutes). Minimum is 60 seconds.

`router_ha_binding_startup_jitter_max`

Type
integer

Default
`60`

Minimum Value
0

Maximum random delay in seconds to add to initial reconciliation start time. This prevents thundering herd issues when multiple agents restart simultaneously (e.g., post-upgrade). A value of 60 means each agent will start reconciliation within 0-60 seconds of startup. Matches l2vni_startup_jitter_max for consistency.

ironic

`auth_strategy`

Type
string

Default
`keystone`

Valid Values
keystone, noauth

Method to use for authentication: noauth or keystone.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
This option is no longer used, please use the [ironic]/auth_type option instead.

`service_type`

Type
string

Default
<None>

The default service_type for endpoint URL discovery.

`service_name`

Type
string

Default
<None>

The default service_name for endpoint URL discovery.

`valid_interfaces`

Type
list

Default
<None>

List of interfaces, in order of preference, for endpoint URL.

`region_name`

Type
string

Default
<None>

The default region_name for endpoint URL discovery.

Table 4: Deprecated Variations

Group	Name
ironic	os_region

`endpoint_override`

Type
string

Default
<None>

Always use this endpoint URL for requests for this client. NOTE: The unversioned endpoint should be specified here; to request a particular API version, use the *version*, *min-version*, and/or *max-version* options.

Table 5: Deprecated Variations

Group	Name
ironic	ironic_url

`version`

Type
string

Default
<None>

Minimum Major API version within a given Major API version for endpoint URL discovery. Mutually exclusive with min_version and max_version

`min_version`

Type
string

Default
<None>

The minimum major version of a given API, intended to be used as the lower bound of a range with max_version. Mutually exclusive with version. If min_version is given with no max_version it is as if max version is latest.

`max_version`

Type
string

Default
<None>

The maximum major version of a given API, intended to be used as the upper bound of a range with min_version. Mutually exclusive with version.

`connect_retries`

Type
integer

Default
<None>

The maximum number of retries that should be attempted for connection errors.

`connect_retry_delay`

Type
floating point

Default
<None>

Delay (in seconds) between two retries for connection errors. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

`status_code_retries`

Type
integer

Default
<None>

The maximum number of retries that should be attempted for retriable HTTP status codes.

Table 6: Deprecated Variations

Group	Name
ironic	max_retries

`status_code_retry_delay`

Type
floating point

Default
<None>

Delay (in seconds) between two retries for retriable status codes. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

Table 7: Deprecated Variations

Group	Name
ironic	retry_interval

`retriable_status_codes`

Type
list

Default
<None>

List of retriable HTTP status codes that should be retried. If not set default to [503]

`interface`

Type
string

Default
<None>

The default interface for endpoint URL discovery.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
Using valid-interfaces is preferrable because it is capable of accepting a list of possible interfaces.

`cafile`

Type
string

Default
<None>

PEM encoded Certificate Authority to use when verifying HTTPs connections.

`certfile`

Type
string

Default
<None>

PEM encoded client certificate cert file

`keyfile`

Type
string

Default
<None>

PEM encoded client certificate key file

`insecure`

Type
boolean

Default
`False`

Verify HTTPS connections.

`timeout`

Type
integer

Default
<None>

Timeout value for http requests

`collect_timing`

Type
boolean

Default
`False`

Collect per-API call timing information.

`split_loggers`

Type
boolean

Default
`False`

Log requests to multiple loggers.

`auth_url`

Type
unknown type

Default
<None>

Authentication URL

`system_scope`

Type
unknown type

Default
<None>

Scope for system operations

`domain_id`

Type
unknown type

Default
<None>

Domain ID to scope to

`domain_name`

Type
unknown type

Default
<None>

Domain name to scope to

`project_id`

Type
unknown type

Default
<None>

Project ID to scope to

`project_name`

Type
unknown type

Default
<None>

Project name to scope to

`project_domain_id`

Type
unknown type

Default
<None>

Domain ID containing project

`project_domain_name`

Type
unknown type

Default
<None>

Domain name containing project

`trust_id`

Type
unknown type

Default
<None>

ID of the trust to use as a trustee use

`user_id`

Type
unknown type

Default
<None>

Users user ID

`username`

Type
unknown type

Default
<None>

Users username

Table 8: Deprecated Variations

Group	Name
ironic	user-name
ironic	user_name

`user_domain_id`

Type
unknown type

Default
<None>

Users domain ID

`user_domain_name`

Type
unknown type

Default
<None>

Users domain name

`password`

Type
unknown type

Default
<None>

Users password

l2vni

`enable_l2vni_trunk_reconciliation`

Type
boolean

Default
`True`

Enable L2VNI trunk port reconciliation based on OVN ha_chassis_group membership. When enabled, the agent will automatically manage trunk subports for network nodes to ensure only required VLANs are trunked to each chassis. This feature creates anchor ports and trunk configurations to bridge overlay networks to physical network infrastructure.

`l2vni_reconciliation_interval`

Type
integer

Default
`180`

Minimum Value
30

Interval in seconds between L2VNI trunk reconciliation runs. Default is 180 seconds (3 minutes).

`l2vni_network_nodes_config`

Type
string

Default
`/etc/neutron/l2vni_network_nodes.yaml`

Path to YAML file containing network node trunk port configuration. Used as fallback when trunk configuration is not available from OVN LLDP data or Ironic. The file should define system_id or hostname, physical_network, and local_link_information for each network node. Network nodes can be identified by either system_id (OVN chassis UUID) or hostname (OVN chassis hostname) for easier configuration.

`l2vni_auto_create_networks`

Type
boolean

Default
`True`

Automatically create Neutron networks for ha_chassis_groups and subport anchors if they do not exist. These networks are used for metadata and modeling, not for passing traffic. If disabled, networks must be pre-created with names matching the expected patterns.

`l2vni_subport_anchor_network`

Type
string

Default
`l2vni-subport-anchor`

Name of the shared network used for all trunk subports. This network is used to signal VLAN bindings to ML2 switch plugins and does not pass actual traffic. Will be auto-created if l2vni_auto_create_networks is enabled.

`l2vni_subport_anchor_network_type`

Type
string

Default
`geneve`

Valid Values
geneve, vxlan

Network type to use for L2VNI anchor networks (both subport anchor and ha_chassis_group networks). These networks are used for metadata and modeling only, not for passing traffic. Must match the overlay network type configured in your environment. If the specified type is not available, network creation will fail with an error rather than falling back to an alternative type.

`l2vni_startup_jitter_max`

Type
integer

Default
`60`

Minimum Value
0

Maximum random delay in seconds to add to initial reconciliation start time. This prevents thundering herd issues when multiple agents restart simultaneously (e.g., post-upgrade). A value of 60 means each agent will start reconciliation within 0-60 seconds of startup.

`enable_l2vni_trunk_reconciliation_events`

Type
boolean

Default
`True`

Enable event-driven L2VNI trunk reconciliation. When enabled, the agent watches OVN Northbound database for localnet port creation and deletion events and triggers immediate reconciliation. This eliminates the stale IDL cache issue and provides sub-second reconciliation latency. Periodic reconciliation still runs as a safety net. Requires enable_l2vni_trunk_reconciliation to be enabled. If disabled, only periodic reconciliation will be used.

`ovn_nb_connection`

Type
list

Default
<None>

OVN Northbound database connection string(s). For HA deployments, specify multiple comma-separated connection strings. Used to query ha_chassis_groups, logical switches, and router ports for L2VNI trunk reconciliation. If not specified, reads from [ovn] ovn_nb_connection (shared with Neutron ML2). Defaults to tcp:127.0.0.1:6641 if neither is configured.

`ovn_sb_connection`

Type
list

Default
<None>

OVN Southbound database connection string(s). For HA deployments, specify multiple comma-separated connection strings. Used to query chassis information and LLDP data for L2VNI trunk reconciliation. If not specified, reads from [ovn] ovn_sb_connection (shared with Neutron ML2). Defaults to tcp:127.0.0.1:6642 if neither is configured.

`ovn_ovsdb_timeout`

Type
integer

Default
<None>

Timeout in seconds for OVN OVSDB connections. If not specified, reads from [ovn] ovsdb_connection_timeout (shared with Neutron ML2). Defaults to 180 if neither is configured.

`ironic_cache_ttl`

Type
integer

Default
`3600`

Minimum Value
300

Time-to-live in seconds for cached Ironic node and port data. Each system_id entry is cached independently and expires after this duration from when it was fetched. This avoids thundering

herd issues when multiple agents are running. A small amount of jitter (10-20%) is automatically added to spread cache refresh times. Default is 3600 seconds (1 hour). Minimum is 300 seconds (5 minutes) to avoid excessive API load.

`ironic_conductor_group`

Type
string

Default
<None>

Ironic conductor group to filter nodes when querying for local_link_information data. This allows the agent to only query nodes managed by a specific conductor group, reducing API load in large deployments. If not specified, all nodes are queried.

`ironic_shard`

Type
string

Default
<None>

Ironic shard to filter nodes when querying for local_link_information data. This allows the agent to only query nodes in a specific shard, reducing API load in large sharded deployments. If not specified, all nodes are queried.

neutron

`service_type`

Type
string

Default
<None>

The default service_type for endpoint URL discovery.

`service_name`

Type
string

Default
<None>

The default service_name for endpoint URL discovery.

`valid_interfaces`

Type
list

Default
<None>

List of interfaces, in order of preference, for endpoint URL.

`region_name`

Type
string

Default
<None>

The default region_name for endpoint URL discovery.

`endpoint_override`

Type
string

Default
<None>

Always use this endpoint URL for requests for this client. NOTE: The unversioned endpoint should be specified here; to request a particular API version, use the *version*, *min-version*, and/or *max-version* options.

`version`

Type
string

Default
<None>

Minimum Major API version within a given Major API version for endpoint URL discovery. Mutually exclusive with min_version and max_version

`min_version`

Type
string

Default
<None>

The minimum major version of a given API, intended to be used as the lower bound of a range with max_version. Mutually exclusive with version. If min_version is given with no max_version it is as if max version is latest.

`max_version`

Type
string

Default
<None>

The maximum major version of a given API, intended to be used as the upper bound of a range with min_version. Mutually exclusive with version.

`connect_retries`

Type
integer

Default
<None>

The maximum number of retries that should be attempted for connection errors.

`connect_retry_delay`

Type
floating point

Default
<None>

Delay (in seconds) between two retries for connection errors. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

`status_code_retries`

Type
integer

Default
<None>

The maximum number of retries that should be attempted for retriable HTTP status codes.

`status_code_retry_delay`

Type
floating point

Default
<None>

Delay (in seconds) between two retries for retriable status codes. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

`retriable_status_codes`

Type
list

Default
<None>

List of retriable HTTP status codes that should be retried. If not set default to [503]

`interface`

Type
string

Default
<None>

The default interface for endpoint URL discovery.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
Using valid-interfaces is preferrable because it is capable of accepting a list of possible interfaces.

`cafile`

Type
string

Default
<None>

PEM encoded Certificate Authority to use when verifying HTTPs connections.

`certfile`

Type
string

Default
<None>

PEM encoded client certificate cert file

`keyfile`

Type
string

Default
<None>

PEM encoded client certificate key file

`insecure`

Type
boolean

Default
`False`

Verify HTTPS connections.

`timeout`

Type
integer

Default
<None>

Timeout value for http requests

`collect_timing`

Type
boolean

Default
`False`

Collect per-API call timing information.

`split_loggers`

Type
boolean

Default
`False`

Log requests to multiple loggers.

`auth_url`

Type
unknown type

Default
<None>

Authentication URL

`system_scope`

Type
unknown type

Default
<None>

Scope for system operations

`domain_id`

Type
unknown type

Default
<None>

Domain ID to scope to

`domain_name`

Type
unknown type

Default
<None>

Domain name to scope to

`project_id`

Type
unknown type

Default
<None>

Project ID to scope to

`project_name`

Type
unknown type

Default
<None>

Project name to scope to

`project_domain_id`

Type
unknown type

Default
<None>

Domain ID containing project

`project_domain_name`

Type
unknown type

Default
<None>

Domain name containing project

`trust_id`

Type
unknown type

Default
<None>

ID of the trust to use as a trustee use

`user_id`

Type
unknown type

Default
<None>

Users user ID

`username`

Type
unknown type

Default
<None>

Users username

Table 9: Deprecated Variations

Group	Name
neutron	user-name
neutron	user_name

user_domain_id

Type
unknown type

Default
<None>

Users domain ID

user_domain_name

Type
unknown type

Default
<None>

Users domain name

password

Type
unknown type

Default
<None>

Users password

Sample Configuration File

The following is a sample ironic-neutron-agent configuration for adaptation and use. For a detailed overview of all available configuration options, refer to *ironic-neutron-agent - Configuration Options*.

The sample configuration can also be viewed in `file form`.

Important

The sample configuration file is auto-generated from networking-baremetal when this documentation is built. You must ensure your version of networking-baremetal matches the version of this documentation.

```
[DEFAULT]

#
# From oslo.log
```

(continues on next page)

(continued from previous page)

```
#

# If set to true, the logging level will be set to DEBUG instead of the␣
↪default
# INFO level. (boolean value)
# Note: This option can be changed without restarting.
#debug = false

# The name of a logging configuration file. This file is appended to any
# existing logging configuration files. For details about logging␣
↪configuration
# files, see the Python logging module documentation. Note that when logging
# configuration files are used then all logging configuration is set in the
# configuration file and other logging configuration options are ignored (for
# example, log-date-format). (string value)
# Note: This option can be changed without restarting.
# Deprecated group/name - [DEFAULT]/log_config
#log_config_append = <None>

# Defines the format string for %%(asctime)s in log records. Default:
# %(default)s . This option is ignored if log_config_append is set. (string
# value)
#log_date_format = %Y-%m-%d %H:%M:%S

# (Optional) Name of log file to send logging output to. If no default is set,
# logging will go to stderr as defined by use_stderr. This option is ignored␣
↪if
# log_config_append is set. (string value)
# Deprecated group/name - [DEFAULT]/logfile
#log_file = <None>

# (Optional) The base directory used for relative log_file  paths. This option
# is ignored if log_config_append is set. (string value)
# Deprecated group/name - [DEFAULT]/logdir
#log_dir = <None>

# Use syslog for logging. Existing syslog format is DEPRECATED and will be
# changed later to honor RFC5424. This option is ignored if log_config_append
# is set. (boolean value)
#use_syslog = false

# Enable journald for logging. If running in a systemd environment you may␣
↪wish
# to enable journal support. Doing so will use the journal native protocol
# which includes structured metadata in addition to log messages.This option␣
↪is
# ignored if log_config_append is set. (boolean value)
#use_journal = false
```

(continues on next page)

(continued from previous page)

```
# Syslog facility to receive log lines. This option is ignored if
# log_config_append is set. (string value)
#syslog_log_facility = LOG_USER

# Use JSON formatting for logging. This option is ignored if log_config_append
# is set. (boolean value)
#use_json = false

# Log output to standard error. This option is ignored if log_config_append is
# set. (boolean value)
#use_stderr = false

# (Optional) Set the 'color' key according to log levels. This option takes
# effect only when logging to stderr or stdout is used. This option is ignored
# if log_config_append is set. (boolean value)
#log_color = false

# The amount of time before the log files are rotated. This option is ignored
# unless log_rotation_type is set to "interval". (integer value)
#log_rotate_interval = 1

# Rotation interval type. The time of the last file change (or the time when
# the service was started) is used when scheduling the next rotation. (string
# value)
# Possible values:
# Seconds - <No description provided>
# Minutes - <No description provided>
# Hours - <No description provided>
# Days - <No description provided>
# Weekday - <No description provided>
# Midnight - <No description provided>
#log_rotate_interval_type = days

# Maximum number of rotated log files. (integer value)
#max_logfile_count = 30

# Log file maximum size in MB. This option is ignored if "log_rotation_type"␣
↪is
# not set to "size". (integer value)
#max_logfile_size_mb = 200

# Log rotation type. (string value)
# Possible values:
# interval - Rotate logs at predefined time intervals.
# size - Rotate logs once they reach a predefined size.
# none - Do not rotate log files.
#log_rotation_type = none

# Format string to use for log messages with context. Used by
```

(continues on next page)

(continued from previous page)

```
# oslo_log.formatters.ContextFormatter (string value)
#logging_context_format_string = %(asctime)s.%(msecs)03d %(process)d
↪%(levelname)s %(name)s [%(global_request_id)s %(request_id)s %(user_
↪identity)s] %(instance)s%(message)s

# Format string to use for log messages when context is undefined. Used by
# oslo_log.formatters.ContextFormatter (string value)
#logging_default_format_string = %(asctime)s.%(msecs)03d %(process)d
↪%(levelname)s %(name)s [-] %(instance)s%(message)s

# Additional data to append to log message when logging level for the message
# is DEBUG. Used by oslo_log.formatters.ContextFormatter (string value)
#logging_debug_format_suffix = %(funcName)s %(pathname)s:%(lineno)d

# Prefix each line of exception output with this format. Used by
# oslo_log.formatters.ContextFormatter (string value)
#logging_exception_prefix = %(asctime)s.%(msecs)03d %(process)d ERROR
↪%(name)s %(instance)s

# Defines the format string for %(user_identity)s that is used in
# logging_context_format_string. Used by oslo_log.formatters.ContextFormatter
# (string value)
#logging_user_identity_format = %(user)s %(project)s %(domain)s %(system_
↪scope)s %(user_domain)s %(project_domain)s

# List of package logging levels in logger=LEVEL pairs. This option is ignored
# if log_config_append is set. (list value)
#default_log_levels = amqp=WARN,boto=WARN,sqlalchemy=WARN,suds=INFO,oslo.
↪messaging=INFO,oslo_messaging=INFO,iso8601=WARN,requests.packages.urllib3.
↪connectionpool=WARN,urllib3.connectionpool=WARN,websocket=WARN,requests.
↪packages.urllib3.util.retry=WARN,urllib3.util.retry=WARN,
↪keystonemiddleware=WARN,routes.middleware=WARN,stevedore=WARN,taskflow=WARN,
↪keystoneauth=WARN,oslo.cache=INFO,oslo_policy=INFO,dogpile.core.dogpile=INFO

# Enables or disables publication of error events. (boolean value)
#publish_errors = false

# The format for an instance that is passed with the log message. (string
# value)
#instance_format = "[instance: %(uuid)s] "

# The format for an instance UUID that is passed with the log message. (string
# value)
#instance_uuid_format = "[instance: %(uuid)s] "

# Interval, number of seconds, of log rate limiting. (integer value)
#rate_limit_interval = 0

# Maximum number of logged messages per rate_limit_interval. (integer value)
```

(continues on next page)

(continued from previous page)

```
#rate_limit_burst = 0

# Log level name used by rate limiting. Logs with level greater or equal to
# rate_limit_except_level are not filtered. An empty string means that all
# levels are filtered. (string value)
# Possible values:
# CRITICAL - <No description provided>
# ERROR - <No description provided>
# INFO - <No description provided>
# WARNING - <No description provided>
# DEBUG - <No description provided>
# '' - <No description provided>
#rate_limit_except_level = CRITICAL

# Enables or disables fatal status of deprecations. (boolean value)
#fatal_deprecations = false


[agent]

#
# From ironic-neutron-agent
#

# Seconds between nodes reporting state to server; should be less than
# agent_down_time, best if it is half or less than agent_down_time. (floating
# point value)
#report_interval = 30

# Log agent heartbeats (boolean value)
#log_agent_heartbeats = false


[baremetal_agent]

#
# From ironic-neutron-agent
#

# Enable HA chassis group alignment reconciliation for router ports on␣
↪networks
# with baremetal external ports. This fixes Launchpad bug #1995078 where
# mismatched HA chassis group priorities between router gateway ports and
# baremetal external ports cause intermittent connectivity issues. When
# enabled, the agent ensures router ports use the same ha_chassis_group as
# baremetal external ports on the same network. (boolean value)
#enable_ha_chassis_group_alignment = true

# Interval in seconds between HA chassis group alignment reconciliation runs.
```

(continues on next page)

(continued from previous page)

```
# This controls how frequently the agent checks for and fixes mismatched HA
# chassis groups. Default is 600 seconds (10 minutes). Minimum is 60 seconds␣
↪to
# avoid excessive API load. (integer value)
# Minimum value: 60
#ha_chassis_group_alignment_interval = 600

# When enabled, HA chassis group alignment only checks resources created or
# updated within the time window specified by
# ha_chassis_group_alignment_window. This reduces reconciliation overhead by
# focusing on recently created resources that may have mismatched HA chassis
# groups. When disabled, performs full reconciliation of all resources on each
# run, which is more thorough but has higher API and database load. (boolean
# value)
#limit_ha_chassis_group_alignment_to_recent_changes_only = true

# Time window in seconds for checking recent resources when
# limit_ha_chassis_group_alignment_to_recent_changes_only is enabled. Default
# is 1200 seconds (20 minutes), which is 2x the default alignment interval.
# Resources created or updated within this window will be checked for HA
# chassis group alignment. Setting to 0 effectively disables windowing even if
# the limit flag is enabled. (integer value)
# Minimum value: 0
#ha_chassis_group_alignment_window = 1200

# Enable router HA binding for router interface ports on networks with
# baremetal nodes. When enabled, the agent automatically binds router␣
↪interface
# ports to the same HA chassis group as the network's external ports, enabling
# proper ARP resolution and connectivity between baremetal nodes and their
# router gateway on VLAN networks. This fixes Launchpad bug #2144458 where
# baremetal nodes experience persistent connectivity failures to their router
# gateway. Uses both event-driven binding (for immediate response) and␣
↪periodic
# reconciliation (for edge cases). (boolean value)
#enable_router_ha_binding = true

# Enable event-driven router HA binding. When enabled, the agent responds
# immediately to HA chassis group creation events by binding router interface
# ports on the affected network. This provides instant connectivity when
# networks are created. Requires enable_router_ha_binding to be enabled. If
# disabled, only periodic reconciliation will be used, which may result in
# connectivity delays until the next reconciliation cycle. (boolean value)
#enable_router_ha_binding_events = true

# Interval in seconds for periodic router HA binding reconciliation. This
# ensures router interface ports are bound to network HA chassis groups even␣
↪if
# events are missed or routers are added after the fact. Default is 600␣
```

(continues on next page)

(continued from previous page)

```
↪seconds
# (10 minutes). Minimum is 60 seconds. (integer value)
# Minimum value: 60
#router_ha_binding_interval = 600

# Maximum random delay in seconds to add to initial reconciliation start time.
# This prevents thundering herd issues when multiple agents restart
# simultaneously (e.g., post-upgrade). A value of 60 means each agent will
# start reconciliation within 0-60 seconds of startup. Matches
# l2vni_startup_jitter_max for consistency. (integer value)
# Minimum value: 0
#router_ha_binding_startup_jitter_max = 60


[ironic]

#
# From ironic-client
#

# DEPRECATED: Method to use for authentication: noauth or keystone. (string
# value)
# Possible values:
# keystone - <No description provided>
# noauth - <No description provided>
# This option is deprecated for removal.
# Its value may be silently ignored in the future.
# Reason: This option is no longer used, please use the [ironic]/auth_type
# option instead.
#auth_strategy = keystone

# The default service_type for endpoint URL discovery. (string value)
#service_type = <None>

# The default service_name for endpoint URL discovery. (string value)
#service_name = <None>

# List of interfaces, in order of preference, for endpoint URL. (list value)
#valid_interfaces = <None>

# The default region_name for endpoint URL discovery. (string value)
# Deprecated group/name - [ironic]/os_region
#region_name = <None>

# Always use this endpoint URL for requests for this client. NOTE: The
# unversioned endpoint should be specified here; to request a particular API
# version, use the `version`, `min-version`, and/or `max-version` options.
# (string value)
# Deprecated group/name - [ironic]/ironic_url
```

(continues on next page)

(continued from previous page)

```
#endpoint_override = <None>

# Minimum Major API version within a given Major API version for endpoint URL
# discovery. Mutually exclusive with min_version and max_version (string␣
↪value)
#version = <None>

# The minimum major version of a given API, intended to be used as the lower
# bound of a range with max_version. Mutually exclusive with version. If
# min_version is given with no max_version it is as if max version is "latest
↪".
# (string value)
#min_version = <None>

# The maximum major version of a given API, intended to be used as the upper
# bound of a range with min_version. Mutually exclusive with version. (string
# value)
#max_version = <None>

# The maximum number of retries that should be attempted for connection␣
↪errors.
# (integer value)
#connect_retries = <None>

# Delay (in seconds) between two retries for connection errors. If not set,
# exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is
# used. (floating point value)
#connect_retry_delay = <None>

# The maximum number of retries that should be attempted for retriable HTTP
# status codes. (integer value)
# Deprecated group/name - [ironic]/max_retries
#status_code_retries = <None>

# Delay (in seconds) between two retries for retriable status codes. If not
# set, exponential retry starting with 0.5 seconds up to a maximum of 60
# seconds is used. (floating point value)
# Deprecated group/name - [ironic]/retry_interval
#status_code_retry_delay = <None>

# List of retriable HTTP status codes that should be retried. If not set
# default to  [503] (list value)
#retriable_status_codes = <None>

# DEPRECATED: The default interface for endpoint URL discovery. (string value)
# This option is deprecated for removal.
# Its value may be silently ignored in the future.
# Reason: Using valid-interfaces is preferrable because it is capable of
# accepting a list of possible interfaces.
```

(continues on next page)

(continued from previous page)

```
#interface = <None>

# PEM encoded Certificate Authority to use when verifying HTTPs connections.
# (string value)
#cafile = <None>

# PEM encoded client certificate cert file (string value)
#certfile = <None>

# PEM encoded client certificate key file (string value)
#keyfile = <None>

# Verify HTTPS connections. (boolean value)
#insecure = false

# Timeout value for http requests (integer value)
#timeout = <None>

# Collect per-API call timing information. (boolean value)
#collect_timing = false

# Log requests to multiple loggers. (boolean value)
#split_loggers = false

# Authentication URL (string value)
#auth_url = <None>

# Scope for system operations (string value)
#system_scope = <None>

# Domain ID to scope to (string value)
#domain_id = <None>

# Domain name to scope to (string value)
#domain_name = <None>

# Project ID to scope to (string value)
#project_id = <None>

# Project name to scope to (string value)
#project_name = <None>

# Domain ID containing project (string value)
#project_domain_id = <None>

# Domain name containing project (string value)
#project_domain_name = <None>

# ID of the trust to use as a trustee use (string value)
```

(continues on next page)

(continued from previous page)

```
#trust_id = <None>

# User's user ID (string value)
#user_id = <None>

# User's username (string value)
# Deprecated group/name - [ironic]/user_name
#username = <None>

# User's domain ID (string value)
#user_domain_id = <None>

# User's domain name (string value)
#user_domain_name = <None>

# User's password (string value)
#password = <None>


[l2vni]

#
# From ironic-neutron-agent
#

# Enable L2VNI trunk port reconciliation based on OVN ha_chassis_group
# membership. When enabled, the agent will automatically manage trunk subports
# for network nodes to ensure only required VLANs are trunked to each chassis.
# This feature creates anchor ports and trunk configurations to bridge overlay
# networks to physical network infrastructure. (boolean value)
#enable_l2vni_trunk_reconciliation = true

# Interval in seconds between L2VNI trunk reconciliation runs. Default is 180
# seconds (3 minutes). (integer value)
# Minimum value: 30
#l2vni_reconciliation_interval = 180

# Path to YAML file containing network node trunk port configuration. Used as
# fallback when trunk configuration is not available from OVN LLDP data or
# Ironic. The file should define system_id or hostname, physical_network, and
# local_link_information for each network node. Network nodes can be␣
↪identified
# by either system_id (OVN chassis UUID) or hostname (OVN chassis hostname)␣
↪for
# easier configuration. (string value)
#l2vni_network_nodes_config = /etc/neutron/l2vni_network_nodes.yaml

# Automatically create Neutron networks for ha_chassis_groups and subport
# anchors if they do not exist. These networks are used for metadata and
```

(continues on next page)

(continued from previous page)

```
# modeling, not for passing traffic. If disabled, networks must be pre-created
# with names matching the expected patterns. (boolean value)
#l2vni_auto_create_networks = true

# Name of the shared network used for all trunk subports. This network is used
# to signal VLAN bindings to ML2 switch plugins and does not pass actual
# traffic. Will be auto-created if l2vni_auto_create_networks is enabled.
# (string value)
#l2vni_subport_anchor_network = l2vni-subport-anchor

# Network type to use for L2VNI anchor networks (both subport anchor and
# ha_chassis_group networks). These networks are used for metadata and␣
↪modeling
# only, not for passing traffic. Must match the overlay network type␣
↪configured
# in your environment. If the specified type is not available, network␣
↪creation
# will fail with an error rather than falling back to an alternative type.
# (string value)
# Possible values:
# geneve - <No description provided>
# vxlan - <No description provided>
#l2vni_subport_anchor_network_type = geneve

# Maximum random delay in seconds to add to initial reconciliation start time.
# This prevents thundering herd issues when multiple agents restart
# simultaneously (e.g., post-upgrade). A value of 60 means each agent will
# start reconciliation within 0-60 seconds of startup. (integer value)
# Minimum value: 0
#l2vni_startup_jitter_max = 60

# Enable event-driven L2VNI trunk reconciliation. When enabled, the agent
# watches OVN Northbound database for localnet port creation and deletion
# events and triggers immediate reconciliation. This eliminates the stale IDL
# cache issue and provides sub-second reconciliation latency. Periodic
# reconciliation still runs as a safety net. Requires
# enable_l2vni_trunk_reconciliation to be enabled. If disabled, only periodic
# reconciliation will be used. (boolean value)
#enable_l2vni_trunk_reconciliation_events = true

# OVN Northbound database connection string(s). For HA deployments, specify
# multiple comma-separated connection strings. Used to query ha_chassis_
↪groups,
# logical switches, and router ports for L2VNI trunk reconciliation. If not
# specified, reads from [ovn] ovn_nb_connection (shared with Neutron ML2).
# Defaults to tcp:127.0.0.1:6641 if neither is configured. (list value)
#ovn_nb_connection = <None>

# OVN Southbound database connection string(s). For HA deployments, specify
```

(continues on next page)

(continued from previous page)

```
# multiple comma-separated connection strings. Used to query chassis
# information and LLDP data for L2VNI trunk reconciliation. If not specified,
# reads from [ovn] ovn_sb_connection (shared with Neutron ML2). Defaults to
# tcp:127.0.0.1:6642 if neither is configured. (list value)
#ovn_sb_connection = <None>

# Timeout in seconds for OVN OVSDB connections. If not specified, reads from
# [ovn] ovsdb_connection_timeout (shared with Neutron ML2). Defaults to 180 if
# neither is configured. (integer value)
#ovn_ovsdb_timeout = <None>

# Time-to-live in seconds for cached Ironic node and port data. Each system_id
# entry is cached independently and expires after this duration from when it
# was fetched. This avoids thundering herd issues when multiple agents are
# running. A small amount of jitter (10-20%%) is automatically added to spread
# cache refresh times. Default is 3600 seconds (1 hour). Minimum is 300␣
↪seconds
# (5 minutes) to avoid excessive API load. (integer value)
# Minimum value: 300
#ironic_cache_ttl = 3600

# Ironic conductor group to filter nodes when querying for
# local_link_information data. This allows the agent to only query nodes
# managed by a specific conductor group, reducing API load in large
# deployments. If not specified, all nodes are queried. (string value)
#ironic_conductor_group = <None>

# Ironic shard to filter nodes when querying for local_link_information data.
# This allows the agent to only query nodes in a specific shard, reducing API
# load in large sharded deployments. If not specified, all nodes are queried.
# (string value)
#ironic_shard = <None>


[neutron]

#
# From ironic-neutron-agent
#

# The default service_type for endpoint URL discovery. (string value)
#service_type = <None>

# The default service_name for endpoint URL discovery. (string value)
#service_name = <None>

# List of interfaces, in order of preference, for endpoint URL. (list value)
#valid_interfaces = <None>
```

(continues on next page)

(continued from previous page)

```
# The default region_name for endpoint URL discovery. (string value)
#region_name = <None>

# Always use this endpoint URL for requests for this client. NOTE: The
# unversioned endpoint should be specified here; to request a particular API
# version, use the `version`, `min-version`, and/or `max-version` options.
# (string value)
#endpoint_override = <None>

# Minimum Major API version within a given Major API version for endpoint URL
# discovery. Mutually exclusive with min_version and max_version (string␣
↪value)
#version = <None>

# The minimum major version of a given API, intended to be used as the lower
# bound of a range with max_version. Mutually exclusive with version. If
# min_version is given with no max_version it is as if max version is "latest
↪".
# (string value)
#min_version = <None>

# The maximum major version of a given API, intended to be used as the upper
# bound of a range with min_version. Mutually exclusive with version. (string
# value)
#max_version = <None>

# The maximum number of retries that should be attempted for connection␣
↪errors.
# (integer value)
#connect_retries = <None>

# Delay (in seconds) between two retries for connection errors. If not set,
# exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is
# used. (floating point value)
#connect_retry_delay = <None>

# The maximum number of retries that should be attempted for retriable HTTP
# status codes. (integer value)
#status_code_retries = <None>

# Delay (in seconds) between two retries for retriable status codes. If not
# set, exponential retry starting with 0.5 seconds up to a maximum of 60
# seconds is used. (floating point value)
#status_code_retry_delay = <None>

# List of retriable HTTP status codes that should be retried. If not set
# default to  [503] (list value)
#retriable_status_codes = <None>
```

(continues on next page)

(continued from previous page)

```
# DEPRECATED: The default interface for endpoint URL discovery. (string value)
# This option is deprecated for removal.
# Its value may be silently ignored in the future.
# Reason: Using valid-interfaces is preferrable because it is capable of
# accepting a list of possible interfaces.
#interface = <None>

# PEM encoded Certificate Authority to use when verifying HTTPs connections.
# (string value)
#cafile = <None>

# PEM encoded client certificate cert file (string value)
#certfile = <None>

# PEM encoded client certificate key file (string value)
#keyfile = <None>

# Verify HTTPS connections. (boolean value)
#insecure = false

# Timeout value for http requests (integer value)
#timeout = <None>

# Collect per-API call timing information. (boolean value)
#collect_timing = false

# Log requests to multiple loggers. (boolean value)
#split_loggers = false

# Authentication URL (string value)
#auth_url = <None>

# Scope for system operations (string value)
#system_scope = <None>

# Domain ID to scope to (string value)
#domain_id = <None>

# Domain name to scope to (string value)
#domain_name = <None>

# Project ID to scope to (string value)
#project_id = <None>

# Project name to scope to (string value)
#project_name = <None>

# Domain ID containing project (string value)
#project_domain_id = <None>
```

(continues on next page)

(continued from previous page)

```
# Domain name containing project (string value)
#project_domain_name = <None>

# ID of the trust to use as a trustee use (string value)
#trust_id = <None>

# User's user ID (string value)
#user_id = <None>

# User's username (string value)
# Deprecated group/name - [neutron]/user_name
#username = <None>

# User's domain ID (string value)
#user_domain_id = <None>

# User's domain name (string value)
#user_domain_name = <None>

# User's password (string value)
#password = <None>
```

4.1.2 Configuration Reference

The following pages describe configuration options that can be used to adjust the neutron ML2 configuration and the baremetal ML2 plug-in and device drivers to your particular situation.

To enable mechanism drivers in the ML2 plug-in, edit the `/etc/neutron/plugins/ml2/ml2_conf.ini` configuration file. For example, this enables the `openvswitch` and `baremetal` mechanism drivers:

```
[ml2]
mechanism_drivers = openvswitch,baremetal
```

To add a device to manage, edit the `/etc/neutron/plugins/ml2/ml2_conf.ini` configuration file. The example below enables devices: `device_a.example.net` and `device_b.example.net`. For each device a separate section in the same configuration file defines the device and driver specific configuration. Please refer to *Device drivers* for details.

```
[networking_baremetal]
enabled_device = device_a.example.net,device_b.example.net
```

Device drivers

The baremetal mechanism ML2 plug-in provides a device driver plug-in interface, this interface can be used to add device (switch) configuration capabilities. The interface uses stevedore for dynamic loading.

Individual drivers may have independent configuration requirements depending on the implementation. *Driver specific options* are documented separately.

Common configuration options for all device drivers

This page describes configuration options that is common to all networking-baremetal device drivers. Individual drivers may have independent configuration requirements depending on the implementation, refer to the device driver specific documentation.

Configuration options

common-example

driver

Type
string

Default
<None>

The driver to use when configuring the device

switch_id

Type
string

Default
<None>

The switch ID, MAC address of the device.

switch_info

Type
string

Default
<None>

Optional string field to be used to store any vendor-specific information.

physical_networks

Type
list

Default
[]

A list of physical networks mapped to this device.

manage_vlans

Type
boolean

Default
True

Set this to False for the device if VLANs should not be create and deleted on the device.

conductor_groups

conductor_groups

Type
list

Default
[]

List of conductor groups this networking-baremetal instance should manage. If empty, all ports will be queried.

networking_baremetal

enabled_devices

Type
list

Default
['common-example', 'netconf-openconfig-example']

This option has a sample default set, which means that its actual default value may vary from the one documented above.

Enabled devices for which the plugin should manageconfiguration. Driver specific configuration for each device must be added in separate sections.

Sample Configuration File

The following is a sample configuration section that would be added to `/etc/neutron/plugins/ml2/ml2_conf.ini`.

The sample configuration can also be viewed in `file form`.

Important

The sample configuration file is auto-generated from networking-baremetal when this documentation is built. You must ensure your version of networking-baremetal matches the version of this documentation.

```
[DEFAULT]


[common-example]

#
# From common-device-driver-opts
#

# The driver to use when configuring the device (string value)
#driver = <None>
```

(continues on next page)

(continued from previous page)

```
# The switch ID, MAC address of the device. (string value)
#switch_id = <None>

# Optional string field to be used to store any vendor-specific information.
# (string value)
#switch_info = <None>

# A list of physical networks mapped to this device. (list value)
#physical_networks =

# Set this to False for the device if VLANs should not be create and deleted␣
↪on
# the device. (boolean value)
#manage_vlans = true


[conductor_groups]

#
# From common-device-driver-opts
#

# List of conductor groups this networking-baremetal instance should manage.␣
↪If
# empty, all ports will be queried. (list value)
#conductor_groups =


[networking_baremetal]

#
# From common-device-driver-opts
#

# Enabled devices for which the plugin should manageconfiguration. Driver
# specific configuration for each device must be added in separate sections.
# (list value)
#
# This option has a sample default set, which means that
# its actual default value may vary from the one documented
# below.
#enabled_devices = common-example,netconf-openconfig-example
```

Available device drivers

Device driver - netconf-openconfig

The `netconf-openconfig` device driver uses the Network Configuration Protocol (NETCONF) and open source vendor-neutral OpenConfig YANG models.

This driver has been tested with the following switch vendor/operating systems:

- Cisco NXOS
- Arista vEOS

Example configuration for Cisco NXOS device:

```
[networking_baremetal]
enabled_devices = nexus.example.net

[nexus.example.net]
driver = netconf-openconfig
device_params = name:nexus
switch_info = nexus
switch_id = 00:53:00:0a:0a:0a
host = nexus.example.net
username = user
key_filename = /etc/neutron/ssh_keys/nexus_sshkey
```

Example configuration for Arista EOS device:

```
[networking_baremetal]
enabled_devices = arista.example.net

[arista.example.net]
driver = netconf-openconfig
device_params = name:default
switch_info = arista
switch_id = 00:53:00:0b:0b:0b
host = arista.example.net
username = user
key_filename = /etc/neutron/ssh_keys/arista_sshkey
```

Configuration options

netconf-openconfig-example

`driver`

Type
string

Default
<None>

The driver to use when configuring the device

`switch_id`

Type
string

Default
<None>

The switch ID, MAC address of the device.

`switch_info`

Type
string

Default
<None>

Optional string field to be used to store any vendor-specific information.

`physical_networks`

Type
list

Default
`[]`

A list of physical networks mapped to this device.

`manage_vlans`

Type
boolean

Default
`True`

Set this to False for the device if VLANs should not be create and deleted on the device.

`network_instance`

Type
string

Default
`default`

Advanced Option
Intended for advanced users and not used by the majority of users, and might have a significant effect on stability and/or performance.

The L2, L3, or L2+L3 forwarding instance to use when defining VLANs on the device.

`port_id_re_sub`

Type
dict

Default
`{'pattern': 'Ethernet', 'repl': 'eth'}`

This option has a sample default set, which means that its actual default value may vary from the one documented above.

Regular expression pattern and replacement string. Some devices do not use the port description from LLDP in Netconf configuration. If the regular expression pattern and replacement string is set the port_id will be modified before passing configuration to the device.

`disabled_properties`

Type
list

Default
`[]`

A list of properties that should not be used, currently only port_mtu is valid

`manage_lacp_aggregates`

Type
boolean

Default
`True`

When set to true the driver will manage LACP aggregates if link_group_information is defined in the binding:profile. When this is false the driver expect the link aggregation to be pre-configured on the device, and only perform vlan plugging.

`link_aggregate_prefix`

Type
string

Default
`Port-Channel`

The device specific prefix used for link-aggregation ports. Common values: po, port-channel or Port-Channel.

`link_aggregate_range`

Type
string

Default
`1000..2000`

Range of link aggregation interface IDs that the driver can use when managing link aggregates.

`host`

Type
string

Default
`device.example.com`

This option has a sample default set, which means that its actual default value may vary from the one documented above.

The hostname or IP address to use for connecting to the netconf device.

`username`

Type
string

Default
`netconf`

This option has a sample default set, which means that its actual default value may vary from the one documented above.

The username to use for SSH authentication.

`port`

Type
integer

Default
`830`

The port to use for connection to the netconf device.

`password`

Type
string

Default
`secret`

This option has a sample default set, which means that its actual default value may vary from the one documented above.

The password used if using password authentication, or the passphrase to use for unlocking keys that require it. (To disable attempting key authentication altogether, set options *allow_agent* and *look_for_keys* to *False*.

`key_filename`

Type
string

Default
`~/.ssh/id_rsa`

Private key filename

`hostkey_verify`

Type
boolean

Default
`True`

Enables hostkey verification from ~/.ssh/known_hosts

`device_params`

Type
dict

Default
`{'name': 'default'}`

ncclient device handler parameters, see ncclient documentation for supported device handlers.

`allow_agent`

Type
boolean

Default
`True`

Enables querying SSH agent (if found) for keys.

`look_for_keys`

Type
boolean

Default
`True`

Enables looking in the usual locations for ssh keys (e.g. `~/.ssh/id_*`)

networking_baremetal

`enabled_devices`

Type
list

Default
`['common-example', 'netconf-openconfig-example']`

This option has a sample default set, which means that its actual default value may vary from the one documented above.

Enabled devices for which the plugin should manageconfiguration. Driver specific configuration for each device must be added in separate sections.

Sample Configuration File

The following is a sample configuration section that would be added to `/etc/neutron/plugins/ml2/ml2_conf.ini`.

The sample configuration can also be viewed in `file form`.

Important

The sample configuration file is auto-generated from networking-baremetal when this documentation is built. You must ensure your version of networking-baremetal matches the version of this documentation.

```
[DEFAULT]


[netconf-openconfig-example]

#
```

(continues on next page)

(continued from previous page)

```
# From netconf-openconfig-driver-opts
#

# The driver to use when configuring the device (string value)
#driver = <None>

# The switch ID, MAC address of the device. (string value)
#switch_id = <None>

# Optional string field to be used to store any vendor-specific information.
# (string value)
#switch_info = <None>

# A list of physical networks mapped to this device. (list value)
#physical_networks =

# Set this to False for the device if VLANs should not be create and deleted␣
↪on
# the device. (boolean value)
#manage_vlans = true

# Regular expression pattern and replacement string. Some devices do not use
# the port description from LLDP in Netconf configuration. If the regular
# expression pattern and replacement string is set the port_id will be␣
↪modified
# before passing configuration to the device. (dict value)
#
# This option has a sample default set, which means that
# its actual default value may vary from the one documented
# below.
#port_id_re_sub = pattern:Ethernet,repl:eth

# A list of properties that should not be used, currently only "port_mtu" is
# valid (list value)
#disabled_properties =

# When set to true the driver will manage LACP aggregates if
# link_group_information is defined in the binding:profile. When this is false
# the driver expect the link aggregation to be pre-configured on the device,
# and only perform vlan plugging. (boolean value)
#manage_lacp_aggregates = true

# The device specific prefix used for link-aggregation ports. Common values:
# "po", "port-channel" or "Port-Channel". (string value)
#link_aggregate_prefix = Port-Channel

# Range of link aggregation interface IDs that the driver can use when␣
↪managing
# link aggregates. (string value)
```

(continues on next page)

(continued from previous page)

```
#link_aggregate_range = 1000..2000

# The hostname or IP address to use for connecting to the netconf device.
# (string value)
#
# This option has a sample default set, which means that
# its actual default value may vary from the one documented
# below.
#host = device.example.com

# The username to use for SSH authentication. (string value)
#
# This option has a sample default set, which means that
# its actual default value may vary from the one documented
# below.
#username = netconf

# The port to use for connection to the netconf device. (integer value)
#port = 830

# The password used if using password authentication, or the passphrase to use
# for unlocking keys that require it. (To disable attempting key␣
↪authentication
# altogether, set options *allow_agent* and *look_for_keys* to `False`. (string
# value)
#
# This option has a sample default set, which means that
# its actual default value may vary from the one documented
# below.
#password = secret

# Private key filename (string value)
#key_filename = ~/.ssh/id_rsa

# Enables hostkey verification from ~/.ssh/known_hosts (boolean value)
#hostkey_verify = true

# ncclient device handler parameters, see ncclient documentation for supported
# device handlers. (dict value)
#device_params = name:default

# Enables querying SSH agent (if found) for keys. (boolean value)
#allow_agent = true

# Enables looking in the usual locations for ssh keys (e.g.
# :file:`~/.ssh/id_*`) (boolean value)
#look_for_keys = true

# The L2, L3, or L2+L3 forwarding instance to use when defining VLANs on the
```

(continues on next page)

(continued from previous page)

```
# device. (string value)
# Advanced Option: intended for advanced users and not used
# by the majority of users, and might have a significant
# effect on stability and/or performance.
#network_instance = default


[networking_baremetal]

#
# From netconf-openconfig-driver-opts
#

# Enabled devices for which the plugin should manageconfiguration. Driver
# specific configuration for each device must be added in separate sections.
# (list value)
#
# This option has a sample default set, which means that
# its actual default value may vary from the one documented
# below.
#enabled_devices = common-example,netconf-openconfig-example
```

L2VNI Mechanism Driver Configuration

Overview

The L2VNI (Layer 2 Virtual Network Identifier) mechanism driver enables baremetal servers to connect to VXLAN and Geneve overlay networks by dynamically creating VLAN segments that bridge the overlay network to the physical network infrastructure.

This driver is essential for deployments where baremetal nodes need to participate in tenant overlay networks alongside virtual machines.

Architecture

How it Works

The L2VNI mechanism driver operates as follows:

1. **Overlay Network Detection**: When a baremetal port is bound to a VXLAN or Geneve network, the driver is triggered.
2. **Dynamic VLAN Allocation**: The driver allocates a dynamic VLAN segment on the specified physical network to carry traffic for the overlay network.
3. **OVN Localnet Port Creation**: If OVN is the backend, the driver creates a localnet port in OVN to bridge the overlay network to the physical VLAN.
4. **Port Binding**: The driver instructs Neutron to continue binding the port using the dynamically allocated VLAN segment.
5. **Traffic Flow**: Traffic flows from the overlay network (VXLAN/Geneve) through the localnet port to the VLAN segment, then to the baremetal server.

```
 Baremetal
  Server

       VLAN 100


  Physical
  Switch

       VLAN 100


 OVN Localnet Port
 (bridges VLANVXLAN)



VXLAN/Geneve
  Overlay
```

Switch Management Integration

The L2VNI mechanism driver works in conjunction with switch management plugins (such as genericswitch) to provide complete end-to-end connectivity for baremetal servers on overlay networks.

Role of Switch Management Plugins

Switch management plugins handle the crucial task of configuring physical network switches to map VNI (VXLAN/Geneve Network Identifier) values to VLAN tags on the physical ports where baremetal servers connect.

When a baremetal port is created or deleted, the following workflow occurs:

1. **L2VNI Driver** (this driver):
    - Allocates a dynamic VLAN segment for the overlay network
    - Creates an OVN localnet port to bridge overlay VLAN
    - Continues the port binding process
2. **Switch Management Plugin** (e.g., genericswitch):
    - Configures the physical switch to map the VLAN to the servers port
    - This is the **final step** in port binding
    - Must be listed **last** in mechanism_drivers

Mechanism Driver Ordering

The order of mechanism drivers in `ml2_conf.ini` is critical:

```
[ml2]
# CORRECT ORDER - switch management MUST be last
mechanism_drivers = ovn,baremetal_l2vni,baremetal,genericswitch

# INCORRECT - will break port binding
mechanism_drivers = ovn,genericswitch,baremetal_l2vni  # WRONG!
```

Why order matters:

- OVN provides the overlay network backend
- baremetal_l2vni allocates the VLAN and creates localnet ports
- baremetal handles standard baremetal port binding
- genericswitch (or other switch management) performs the **final** switch configuration step

If the switch management plugin runs too early, it wont have the correct VLAN information allocated by baremetal_l2vni, causing port binding to fail.

Requirements

- OpenStack Neutron with ML2 plugin
- OVN (Open Virtual Network) backend (**required** - this driver requires OVN)
- Physical network switches configured for VLAN trunking
- Switch management ML2 plugin (e.g., genericswitch) for VNIVLAN mapping
- Baremetal nodes with appropriate VLAN configuration

Configuration

Enabling the Driver

Edit `/etc/neutron/plugins/ml2/ml2_conf.ini` and add `baremetal_l2vni` to the list of mechanism drivers:

```
[ml2]
mechanism_drivers = ovn,baremetal_l2vni,baremetal,genericswitch
```

Important

Driver order is critical:

- `ovn` must be first (provides overlay network backend)
- `baremetal_l2vni` allocates VLANs and creates localnet ports
- `baremetal` handles standard baremetal port binding
- `genericswitch` (or other switch management) must be **last** to perform final switch configuration

Configuration Options

Add a `[baremetal_l2vni]` section to your configuration file:

> **Note**
>
> A complete configuration example is available at `l2vni-example.ini`

```
[baremetal_l2vni]
# Enable automatic creation of OVN localnet ports (default: True)
create_localnet_ports = True

# Default physical network for baremetal ports (optional)
# If not set, ports must specify physical_network in binding profile
default_physical_network = physnet1
```

Configuration Parameters

`create_localnet_ports`

Type: Boolean

Default: `True`

Description: Automatically create OVN localnet ports to bridge VXLAN/Geneve overlay networks to physical networks.

When to use True (default):

- Direct VLAN-to-VXLAN fabric attachment scenarios
- When using ML2 plugin for direct attachment to a VLAN-to-VXLAN fabric
- The OVN localnet ports enable the overlayphysical network translation

When to use False:

- Pure EVPN deployments where Neutron is responsible for ensuring attachment to the remote network infrastructure through tunnels rather than through localnet ports in OVN
- When localnet ports are managed externally

> **Note**
>
> If youre using EVPN where network attachment is handled via tunnels, you likely want to set this to `False` since localnet ports are not needed for that architecture.

`default_physical_network`

Type: String

Default: `None`

Description: Default physical network name to use for baremetal L2VNI bindings when the port binding profile does not specify a `physical_network`. If not set and the port lacks `physical_network` in its binding profile, port binding will fail.

Port Binding Profile

When creating baremetal ports, you can specify the physical network in the binding profile:

```
openstack port create \
  --network overlay-network \
  --vnic-type baremetal \
  --binding-profile physical_network=physnet1 \
  baremetal-port
```

If `default_physical_network` is configured, the binding profile is optional.

Network Configuration

Physical Networks

Ensure your physical networks are properly configured in ML2:

```
[ml2_type_vlan]
network_vlan_ranges = physnet1:100:200
```

On each chassis (compute/network node), configure OVN bridge mappings:

```
ovs-vsctl set Open_vSwitch . \
  external-ids:ovn-bridge-mappings=physnet1:br-provider
```

Router Configuration

When baremetal networks are attached to Neutron routers, ensure the router has an external gateway configured for proper routing behavior. The driver automatically configures router gateway chassis bindings when necessary.

Deployment Guide

Step 1: Enable the Mechanism Driver

Edit `/etc/neutron/plugins/ml2/ml2_conf.ini`:

```
[ml2]
mechanism_drivers = ovn,baremetal_l2vni,baremetal,genericswitch
type_drivers = flat,vlan,vxlan,geneve
project_network_types = vxlan

[baremetal_l2vni]
create_localnet_ports = True
default_physical_network = physnet1
```

Important

Ensure mechanism drivers are in the correct order: OVN, baremetal_l2vni, baremetal, genericswitch (or other switch management plugin last).

Step 2: Configure Physical Networks

Ensure VLAN ranges are configured:

```
[ml2_type_vlan]
network_vlan_ranges = physnet1:100:200
```

Step 3: Configure OVN Bridge Mappings

On each chassis that will handle baremetal traffic:

```
ovs-vsctl set Open_vSwitch . \
  external-ids:ovn-bridge-mappings=physnet1:br-provider
```

Step 4: Restart Neutron Server

```
systemctl restart neutron-server
```

Step 5: Create Overlay Network

Create a tenant overlay network. You must explicitly specify the network type as VXLAN or Geneve (the only supported types for this driver):

```
openstack network create \
  overlay-network

openstack subnet create \
  --network overlay-network \
  --subnet-range 192.168.100.0/24 \
  overlay-subnet
```

Warning

Do not use provider networks (`--provider-physical-network`, with this driver. Provider networks are intended to be pre-configured for direct attachment, where as this model and interaciton requires additional confiuration and actions to occur.

Note

Only VXLAN and Geneve network types are supported. If your default network type is configured to something else (e.g., VLAN or flat), then this plugin will not work as intended.

Step 6: Create Baremetal Port

Create a baremetal port on the overlay network:

Note

This step is intended for manually triggering the binding logic which demonstrates the mechanism driver creating lower binding segment. In normal usage flow of this plugin, Ironic manages the binding profile and vnic type attributes of ports.

```
openstack port create \
  --network overlay-network \
  --vnic-type baremetal \
  --binding-profile physical_network=physnet1 \
  baremetal-port
```

The driver will automatically:

- Allocate a dynamic VLAN segment (e.g., VLAN 150) on physnet1
- Create an OVN localnet port to bridge VXLAN VLAN
- Bind the port using the VLAN segment

Troubleshooting

Port Binding Fails

Symptom: Port remains in `DOWN` state or binding fails.

Possible Causes:

1. **Missing physical_network**: Port binding profile doesnt specify `physical_network` and no `default_physical_network` is configured.

   Solution: Either specify physical_network in binding profile or configure `default_physical_network`.

2. **Physical network not found**: No chassis has the specified physical network in bridge-mappings.

   Solution: Check logs for error message and verify OVN bridge-mappings configuration on all chassis.

3. **VLAN exhaustion**: No available VLANs in the configured range.

   Solution: Expand VLAN range in `ml2_type_vlan` configuration.

Localnet Port Not Created

Symptom: Port binds but traffic doesnt flow.

Possible Causes:

1. **Localnet creation disabled**: `create_localnet_ports = False`

   Solution: Set `create_localnet_ports = True` or manage localnet ports externally.

2. **OVN not available**: Driver cannot connect to OVN.

   Solution: Check Neutron logs for OVN connection errors. Verify OVN mechanism driver is loaded.

3. **Chassis without physnet**: No chassis has the physical network configured.

   Solution: Configure `ovn-bridge-mappings` on at least one chassis.

Router Attachment Breaks Connectivity

Symptom: Adding a router to the network breaks baremetal connectivity.

Possible Causes:

1. **Router without external gateway**: Router has no gateway port, causing OVN to remove external port bindings.

   Solution: Configure an external gateway for the router, or ensure the gateway interface is up.

2. **Gateway chassis mismatch**: Router gateway is on a different chassis than the localnet port.

   Solution: The driver handles this automatically. Check logs for gateway chassis binding messages.

Checking Logs

Enable debug logging for detailed information:

```
[DEFAULT]
debug = True
```

Check Neutron server logs:

```
journalctl -u neutron-server -f
```

Look for messages containing:

- `L2vniMechanismDriver` - General driver operations
- `localnet port` - Localnet port creation/deletion
- `physical_network` - Physical network validation
- `allocate dynamic segment` - VLAN segment allocation

Verifying OVN State

Check OVN Northbound database:

```
# List logical switches and ports
ovn-nbctl show

# Look for localnet ports (format: neutron-<network-id>-localnet-<physnet>)
ovn-nbctl list Logical_Switch_Port | grep localnet
```

Check OVN Southbound database:

```
# List chassis and their bridge-mappings
ovn-sbctl list Chassis

# Check port bindings
ovn-sbctl list Port_Binding
```

Advanced Topics

Multiple Physical Networks

You can use different physical networks for different ports:

```
openstack port create \
  --network overlay-network \
  --vnic-type baremetal \
  --binding-profile physical_network=physnet1 \
  port-on-physnet1

openstack port create \
  --network overlay-network \
  --vnic-type baremetal \
  --binding-profile physical_network=physnet2 \
  port-on-physnet2
```

The driver will create separate VLAN segments and localnet ports for each physical network.

VLAN Segment Reuse

The driver is idempotent - if a VLAN segment already exists for a given overlay network + physical network combination, it will reuse the existing segment rather than allocating a new one.

Segment Cleanup

When the last baremetal port using a dynamic VLAN segment is deleted or unbound, the driver automatically:

1. Removes the OVN localnet port
2. Releases the dynamic VLAN segment back to the pool

Performance Considerations

VLAN Pool Sizing

Plan your VLAN ranges carefully. Each overlay network that has baremetal ports on a given physical network requires one VLAN from the pool.

For example, with 100 tenant overlay networks and baremetal nodes on 2 physical networks, you need up to 200 VLANs.

OVN Database Load

The driver queries the OVN Southbound database to validate physical network availability. In very large deployments (1000+ chassis), this query may add latency to port binding operations.

See Also

- *Configuration Reference* - ML2 Plugin Configuration
- *Contributing* - Contributing Guide
- OpenStack Neutron Documentation: https://docs.openstack.org/neutron/
- OVN Documentation: https://www.ovn.org/

CHAPTER

FIVE

ADMINISTRATOR GUIDE

5.1 Administrator Guide

This guide provides information for deploying, configuring, and operating networking-baremetal in production environments.

5.1.1 HA Chassis Group Alignment

Overview

The HA Chassis Group Alignment feature addresses connectivity issues that can occur in OVN deployments with baremetal nodes when router gateway ports and baremetal external ports have mismatched HA chassis group priorities.

This feature implements automatic reconciliation to ensure router ports use the same HA chassis group configuration as baremetal external ports on the same network, eliminating intermittent connectivity failures.

Problem Statement

In OpenStack deployments using OVN with baremetal nodes, external connectivity can fail intermittently due to a configuration mismatch. This occurs when:

1. A baremetal node has an external port (`device_owner=baremetal:none`) on a provider network
2. A router is attached to the same network via a router interface port
3. OVN assigns different HA chassis groups with different priorities to the baremetal external port and the router gateway port
4. The active chassis for the baremetal port differs from the active chassis for the router port

When this mismatch occurs, traffic routing becomes inconsistent, causing baremetal nodes to lose external connectivity intermittently as different chassis believe they are the active gateway.

This issue is tracked in Launchpad as bug #1995078: https://bugs.launchpad.net/neutron/+bug/1995078

Solution

The ironic-neutron-agent now includes a periodic reconciliation loop that:

1. Discovers all networks with baremetal external ports
2. Identifies the HA chassis group used by baremetal ports on each network
3. Finds router interface ports on the same networks

4. Updates router ports to use the same HA chassis group as baremetal ports
5. Only processes networks managed by the agent instance (via hash ring)

This ensures consistent HA chassis group configuration across all ports on networks with baremetal nodes, preventing the priority mismatch that causes connectivity failures.

Configuration

The feature is controlled by options in the `[baremetal_agent]` section of the agent configuration file (typically `/etc/neutron/ironic_neutron_agent.ini`).

Enable/Disable

```
[baremetal_agent]
# Enable HA chassis group alignment reconciliation
# Default: True
enable_ha_chassis_group_alignment = True
```

Set to `False` to disable the feature if you are not experiencing the connectivity issue or if you have resolved it through other means.

Reconciliation Interval

```
[baremetal_agent]
# Interval in seconds between alignment reconciliation runs
# Default: 600 (10 minutes)
# Minimum: 60
ha_chassis_group_alignment_interval = 600
```

Controls how frequently the agent checks for and fixes HA chassis group mismatches. The default of 10 minutes provides a balance between:

- Timely detection and correction of mismatches
- Minimal impact on Neutron API and OVN database load

For deployments with frequent network topology changes, you may want to reduce this interval. For stable deployments, you can increase it to reduce overhead.

Time Window Filtering

```
[baremetal_agent]
# Only check recently created/updated resources
# Default: True
limit_ha_chassis_group_alignment_to_recent_changes_only = True

# Time window in seconds for "recent" resources
# Default: 1200 (20 minutes, 2x the alignment interval)
# Minimum: 0
ha_chassis_group_alignment_window = 1200
```

When enabled, reconciliation only examines ports that have been created or updated within the specified time window. This significantly reduces API and database load in large deployments by focusing on resources most likely to have mismatches (newly created ports).

When to disable: Set `limit_ha_chassis_group_alignment_to_recent_changes_only = False` if you:

- Want to perform full reconciliation on every run
- Are recovering from a period where the agent was disabled
- Suspect existing ports have mismatches that need correction

Operational Considerations

Multi-Agent Deployments

In deployments with multiple ironic-neutron-agent instances:

- Each agent uses a distributed hash ring to determine which networks it manages
- Only the responsible agent will reconcile a given network
- This prevents duplicate work and API contention
- If an agent fails, other agents will automatically take over its networks

Monitoring

The agent logs alignment activities at the INFO level:

```
INFO ... Started HA chassis group alignment reconciliation loop
         (interval: 600s, first run in 42s)
INFO ... Updating router port <uuid> HA chassis group from <old> to <new>
         (network <uuid>)
INFO ... Successfully updated router port <uuid> HA chassis group
```

Failed updates are logged at ERROR level with full exception details.

Performance Impact

The reconciliation loop has minimal performance impact:

- **Default configuration:** Queries Neutron for baremetal ports every 10 minutes
- **With windowing enabled (default):** Only checks recently updated ports
- **Uses existing OVN connections:** Reuses connections from L2VNI trunk manager if available
- **Distributed load:** Multiple agents split work via hash ring

In a deployment with 1000 baremetal nodes and default settings:

- First Neutron query returns ~1000 ports
- With 20-minute window, ~50 ports processed per reconciliation (assuming 5% churn rate)
- Per-network processing: 1-2 additional Neutron queries, 2-3 OVN queries
- Total: ~100-150 API calls every 10 minutes across all agents

Troubleshooting

Verifying the Feature is Running

Check agent logs for startup message:

```
$ grep "HA chassis group alignment" /var/log/neutron/ironic-neutron-agent.log
INFO ... HA chassis group alignment reconciliation enabled
INFO ... Started HA chassis group alignment reconciliation loop
         (interval: 600s, first run in 42s)
```

Checking for Mismatches

If you suspect an alignment issue:

1. Identify the affected network and baremetal ports
2. Check OVN for the HA chassis group on baremetal ports:

   ```
   $ ovn-nbctl lsp-get-ha-chassis-group <port-uuid>
   ```

3. Check router ports on the same network:

   ```
   $ ovn-nbctl lrp-get-ha-chassis-group lrp-<router-port-uuid>
   ```

4. If different, the next reconciliation cycle will align them (check logs)

Forcing Immediate Reconciliation

To trigger reconciliation without waiting for the interval:

1. Restart the ironic-neutron-agent
2. The first reconciliation runs within 60 seconds (with random jitter)

Alternatively, temporarily reduce the interval:

```
$ openstack-config --set /etc/neutron/ironic_neutron_agent.ini \
    baremetal_agent ha_chassis_group_alignment_interval 60
$ systemctl restart ironic-neutron-agent
```

Related Features

This feature complements the L2VNI trunk reconciliation feature:

- **L2VNI reconciliation:** Manages VLAN trunk configurations for network nodes
- **HA alignment:** Ensures consistent HA chassis group configuration

Both features can be enabled independently and run on separate schedules.

References

- Launchpad Bug #1995078: https://bugs.launchpad.net/neutron/+bug/1995078
- OVN HA Chassis Groups: https://www.ovn.org/support/dist-docs/ovn-nb.5.html
- ironic-neutron-agent: https://docs.openstack.org/networking-baremetal/

5.1.2 Router HA Binding for VLAN Networks

Overview

The Router HA Binding feature ensures router interface ports are bound to the same HA chassis group as the networks external ports, enabling proper connectivity between baremetal nodes and routers on VLAN networks.

This feature uses event-driven binding when HA chassis groups are created, plus periodic reconciliation.

Without this feature, baremetal nodes on VLAN networks cannot communicate with their router gateway because the routers internal interface port (Logical Router Port) is not bound to any chassis. When a baremetal node tries to ARP for the router IP, no chassis responds because the router interface port has no HA chassis group set.

Implementation

The ironic-neutron-agent now includes a **RouterHABindingManager** that automatically binds router interface ports to network HA chassis groups using a dual approach:

Event-Driven Binding

The agent monitors OVNs `HA_Chassis_Group` table for network-level groups. When a network HA chassis group is created or updated:

1. `HAChassisGroupNetworkEvent` fires immediately
2. Agent finds all router interface ports on that network
3. Binds each router port to the networks HA chassis group
4. Router can now respond to ARP requests on the physical VLAN network

This provides **immediate** connectivity with no delay.

Periodic Reconciliation

A periodic reconciliation loop (default: 10 minutes) ensures eventual consistency by:

1. Discovering all networks with HA chassis groups
2. Finding router interface ports on those networks
3. Binding any unbound or incorrectly bound router ports
4. Only processing networks managed by this agent (via hash ring)

This catches edge cases such as:

- Routers added to existing networks (no event fires)
- Missed events (agent down during HA chassis group creation)
- Manual changes to router port configuration
- Race conditions or out-of-order event processing

Configuration

The feature is controlled by options in the `[baremetal_agent]` section of the agent configuration file (typically `/etc/neutron/ironic_neutron_agent.ini`).

Enable/Disable

```
[baremetal_agent]
# Enable router HA binding for VLAN networks
# Default: True
enable_router_ha_binding = True
```

Set to `False` to disable the feature if you are not using baremetal nodes on VLAN networks with routers.

Event-Driven Binding

```
[baremetal_agent]
# Enable event-driven router HA binding
# Default: True
enable_router_ha_binding_events = True
```

When enabled, the agent responds immediately to HA chassis group creation events by binding router interface ports on the affected network. This provides instant connectivity when networks are created.

Set to `False` to disable event-driven binding and rely only on periodic reconciliation. This may result in connectivity delays until the next reconciliation cycle (default: 10 minutes).

Note: Requires `enable_router_ha_binding = True` to have any effect.

Reconciliation Interval

```
[baremetal_agent]
# Interval in seconds between periodic reconciliation runs
# Default: 600 (10 minutes)
# Minimum: 60
router_ha_binding_interval = 600
```

Controls how frequently the agent performs full reconciliation.

Startup Jitter

```
[baremetal_agent]
# Maximum random delay for initial reconciliation start
# Default: 60 seconds
# Minimum: 0
router_ha_binding_startup_jitter_max = 60
```

Adds random delay (0 to max seconds) before first reconciliation run to prevent thundering herd when multiple agents restart simultaneously. A value of 60 means each agent starts reconciliation within 0-60 seconds of startup.

Operational Considerations

Multi-Agent Deployments

In deployments with multiple ironic-neutron-agent instances:

- Each agent uses a distributed hash ring to determine which networks it manages

- Only the responsible agent will reconcile a given network
- This prevents duplicate work and API contention
- If an agent fails, other agents will automatically take over its networks

Monitoring

The agent logs binding activities at the INFO level:

```
INFO ... Router HA binding enabled, initializing manager
INFO ... Started router HA binding reconciliation loop
         (interval: 600s, first run in 42s)
INFO ... Registered OVN event handler for HA chassis group network events
INFO ... Network HA chassis group ... created/updated for network ...,
         triggering router interface binding
INFO ... Updated router port <uuid> HA chassis group from <old> to <new>
         (network <uuid>)
INFO ... Router HA binding reconciliation complete: processed N networks,
         updated M router ports
```

Failed updates are logged at ERROR level with full exception details.

Performance Impact

The feature has minimal performance impact:

- **Event-driven binding:** Immediate response with no periodic overhead
- **Periodic reconciliation:** Runs every 10 minutes (configurable)
- **Idempotent operations:** Most checks are already correct (cheap)
- **Uses existing OVN connections:** Reuses connections from L2VNI trunk manager
- **Distributed load:** Multiple agents split work via hash ring

In a deployment with 100 networks with HA chassis groups:

- Event-driven: 1-2 Neutron queries, 1-2 OVN updates per HA chassis group creation
- Periodic: Scans all HA chassis groups, queries router ports per network
- Per-network processing: 1 Neutron query, 1-2 OVN operations
- Total periodic: ~100-200 operations every 10 minutes across all agents

Since events handle 99% of cases immediately, periodic reconciliation overhead is minimal.

Troubleshooting

Verifying the Feature is Running

Check agent logs for startup messages:

```
$ grep "router HA binding" /var/log/neutron/ironic-neutron-agent.log
INFO ... Router HA binding enabled, initializing manager
INFO ... Started router HA binding reconciliation loop
```

(continues on next page)

(continued from previous page)

```
        (interval: 600s, first run in 42s)
INFO ... Registered OVN event handler for HA chassis group network events
```

Checking Router Interface Binding

If baremetal nodes cannot reach their gateway:

1. **Verify the network has an HA chassis group:**

```
$ sudo ovn-nbctl ha-chassis-group-list | grep neutron-<network-id>
c18ab533-... (neutron-a72fd10e-...)
    5668117a-... (3773bfbe-...) priority 1
```

2. **Check if router interface port is bound:**

```
$ ROUTER_PORT_ID=$(openstack port list --network <network-id> \
    --device-owner network:router_interface -c ID -f value)
$ sudo ovn-nbctl get Logical_Router_Port lrp-$ROUTER_PORT_ID ha_chassis_
↪group

# Should show UUID, not []
c18ab533-09b9-48fc-8acd-9407bd3f25d2
```

3. **If router port shows []:** Wait for next reconciliation or check logs for errors

Forcing Immediate Reconciliation

To trigger reconciliation without waiting:

```
$ systemctl restart ironic-neutron-agent

# First reconciliation runs within 60 seconds (random jitter)
```

See Also

- *L2VNI Trunk Reconciliation Configuration* - L2VNI Trunk Reconciliation Configuration
- *Configuration Reference* - Agent Configuration Reference

5.1.3 L2VNI Trunk Reconciliation Configuration

Overview

The L2VNI trunk reconciliation feature enables automatic management of trunk ports on network nodes to bridge OVN overlay networks to physical network infrastructure. This feature is essential for deployments where baremetal nodes need to connect to overlay networks through network nodes acting as gateways.

What Problem Does This Solve?

In deployments using the L2VNI mechanism driver, baremetal nodes connect to VXLAN/Geneve overlay networks via VLAN segments on physical networks. For this to work, network nodes must:

1. Have trunk ports configured with the correct VLAN subports
2. Keep those VLANs synchronized with active overlay networks
3. Clean up VLANs when overlay networks are removed

Manual management of these trunk ports becomes impractical at scale, especially when:

- Multiple network nodes form ha_chassis_groups for high availability
- Overlay networks are frequently created and deleted
- Router migrations cause chassis membership changes
- Network nodes are added or removed from the infrastructure

The trunk reconciliation feature automates this entire process using two complementary mechanisms that can be used independently or together:

1. **OVN Event-Driven Reconciliation**: Watches OVN database for localnet port changes and triggers immediate reconciliation
2. **Periodic Reconciliation**: Runs at regular intervals as a safety net to ensure eventual consistency

Recommended for production: Enable both mechanisms. Event-driven provides immediate response to changes, while periodic reconciliation ensures eventual consistency even if events are missed or the agent is restarted.

Architecture

Reconciliation Mechanisms

The agent uses two reconciliation mechanisms that can be used independently or together.

1. Event-Driven L2VNI Trunk Reconciliation (Default: Enabled)

When enabled (`enable_l2vni_trunk_reconciliation_events = True`), the agent watches OVN Northbound database for localnet port creation and deletion events. This provides immediate reconciliation when the L2VNI mechanism driver creates or deletes localnet ports during baremetal port binding operations. This eliminates stale IDL cache issues and enables targeted updates for specific VLANs without scanning all trunk ports.

How It Works:

1. L2VNI mechanism driver creates localnet port in OVN Northbound DB
2. OVN sends notification to all connected IDL clients
3. Agents IDL processes the notification via `idl.run()`
4. `LocalnetPortEvent.matches()` filters for L2VNI localnet ports and hash ring ownership
5. `LocalnetPortEvent.run()` extracts network ID, physnet, and VLAN ID from the event
6. Agent performs **targeted reconciliation** for that specific VLAN:
    - Ensures infrastructure networks exist

- Queries overlay segment information (VNI) for the network
- Finds/creates trunks for chassis with the physnet
- Adds or removes only the specific VLAN subport with VNI in binding profile (idempotent)
- Skips scanning all other VLANs on the trunk

This targeted approach is significantly faster than full reconciliation, especially on trunks with many VLANs. The event handler is registered at agent startup and watches continuously for localnet port changes.

2. Periodic Reconciliation (Default: Enabled)

Periodic reconciliation runs at the configured interval (`l2vni_reconciliation_interval`) as a safety net to catch any missed events, handle agent restarts, and ensure eventual consistency.

Operating Modes

The agent supports three operating modes:

1. **Both Enabled (Recommended for Production)**

```
[l2vni]
enable_l2vni_trunk_reconciliation = True
enable_l2vni_trunk_reconciliation_events = True
```

   - Event-driven reconciliation provides immediate response
   - Periodic reconciliation ensures eventual consistency
   - Best reliability and performance

2. **Event-Driven Only (Testing/Advanced)**

```
[l2vni]
enable_l2vni_trunk_reconciliation = False
enable_l2vni_trunk_reconciliation_events = True
```

   - Only event-driven reconciliation runs
   - No periodic safety net
   - Useful for testing event-driven behavior in isolation
   - Lower overhead but no eventual consistency guarantee

3. **Periodic Only**

```
[l2vni]
enable_l2vni_trunk_reconciliation = True
enable_l2vni_trunk_reconciliation_events = False
```

   - Only periodic reconciliation runs
   - Reconciliation occurs at configured interval

How It Works

The ironic-neutron-agent runs a reconciliation process that:

1. **Discovers Network Nodes**: Identifies chassis that are members of OVN ha_chassis_groups by querying the OVN Northbound database.
2. **Creates Trunk Infrastructure**: For each (chassis, physical_network) combination, ensures:
   - An anchor port exists on the ha_chassis_group network
   - A trunk port is created using that anchor port
   - The trunk is properly named for tracking
3. **Calculates Required VLANs**: Analyzes OVN state to determine which VLANs are needed on each trunk:
   - Queries logical router ports for gateway chassis assignments
   - Identifies overlay networks attached to those routers
   - Finds the dynamic VLAN segment allocated for each overlay network
   - Retrieves overlay segment information (VNI) for L2VNI mapping
4. **Reconciles Subports**: Ensures each trunk has exactly the right set of VLAN subports:
   - Adds missing subports for new overlay networks with VNI in binding profile
   - Removes subports for deleted overlay networks
   - Updates subport binding profiles with switch connection information
5. **Cleans Up Orphans**: Removes infrastructure for deleted network nodes:
   - Deletes trunks for chassis no longer in ha_chassis_groups
   - Removes anchor ports and subports
   - Cleans up networks for deleted ha_chassis_groups

Components and Data Flow

```
                OVN Northbound DB
- HA Chassis Groups
- Logical Router Ports
- Gateway Chassis Assignments


                Queries


         ironic-neutron-agent


   L2VNI Trunk Manager
```

(continues on next page)

(continued from previous page)

```
   1. Discover chassis in ha_groups
   2. Calculate required VLANs
   3. Reconcile trunk subports
   4. Cleanup orphaned resources




        Neutron API calls


                   Neutron Server
 - Creates/deletes trunk ports
 - Manages subports
 - Coordinates with ML2 plugin


                ML2 mechanism drivers


             Physical Switch Plugins
 (e.g., genericswitch)
 - Configures trunk ports on physical switches
 - Maps VLANs to network node ports
```

Infrastructure Objects

HA Chassis Group Networks

For each ha_chassis_group in OVN, the reconciliation process creates a Neutron network named `l2vni-ha-group-{group_uuid}`. These networks are used to host anchor ports and provide network context for trunk ports.

Anchor Ports

Each (chassis, physical_network) combination gets an anchor port named `l2vni-anchor-{system_id}-{physnet}`. The anchor port:

- Is created on the ha_chassis_group network
- Has device_owner `baremetal:l2vni_anchor`
- Contains binding profile with system_id, physical_network, and local_link_information
- Serves as the parent port for the trunk
- Does not have binding:host_id set (not required for switch configuration)

The local_link_information in the anchor ports binding profile is used by networking-generic-switch to configure the physical switch port when subports are added to the trunk. Note that local_link_information is a list containing one or more link connection dictionaries:

- **Single link**: `[{switch_id: ..., port_id: ..., switch_info: ...}]`

- **Multiple links (LAG/bonding)**: `[{...}, {...}, ...]`

When multiple links are configured (from OVN LLDP, Ironic, or YAML config), they are all aggregated into this list, enabling ML2 mechanism drivers to configure link aggregation groups on the physical switch.

Trunk Ports

Trunks are named `l2vni-trunk-{system_id}-{physnet}` and use the anchor port as their parent. The trunk carries multiple VLAN-tagged subports.

Subports

Each subport represents one overlay networks VLAN segment:

- Named `l2vni-subport-{system_id}-{physnet}-vlan{vlan_id}`
- Created on the subport anchor network (`l2vni-subport-anchor` by default)
- Has device_owner `baremetal:l2vni_subport`
- Has binding:host_id set to the chassis hostname for proper ML2 binding
- Segmentation type is always `vlan`
- Contains `binding:profile` with:
  - `physical_network`: The physical network name
  - `segment_id`: The Neutron VLAN segment UUID
  - `vni`: The overlay segment ID (VXLAN/Geneve VNI) for L2VNI mapping

The segment_id enables ML2 mechanism drivers to use segment-based cleanup logic instead of parsing switch port configurations, allowing static trunk port configurations. The VNI enables complete VLAN-to-VNI mappings on physical switches for EVPN/VXLAN bridging.

Note: ML2 mechanism drivers use the parent ports (anchor ports) local_link_information when configuring the physical switch, so subports do not need local_link_information in their binding profile.

Subport Anchor Network

A shared network (default name: `l2vni-subport-anchor`) hosts all subports across all trunks. This network is used to signal VLAN bindings to ML2 switch plugins and does not pass actual traffic.

Switch Connection Information

The reconciliation process attempts to discover switch connection information for each trunk from multiple sources, in priority order:

1. **OVN LLDP Data**: Extracted from OVN Southbound Port table external_ids (`lldp_chassis_id`, `lldp_port_id`, `lldp_system_name`). If multiple ports on the chassis have LLDP data for the same bridge, all links are aggregated for LAG/bonding support.
2. **Ironic Node Data**: Retrieved from Ironic port local_link_information for nodes matching the chassis system-id. **Cached per system_id** with configurable TTL to reduce API load. If multiple Ironic ports have the same physical_network, all links are aggregated for LAG/bonding support.

   Queries use field filtering (only fetching uuid, properties, physical_network, and local_link_information) for optimal performance.

   Optional filtering by conductor_group and shard reduces query scope in large deployments.

3. **YAML Configuration File**: Fallback configuration from `l2vni_network_nodes_config` file. Supports both single-link and multi-link (LAG/bonding) configurations.

Performance Note:

Ironic data is cached per system_id with individual expiration times (TTL + jitter). In a deployment with 1000 nodes, this reduces API calls from ~16,000 per reconciliation to ~2 (when cached), while still refreshing stale data automatically.

This information is included in subport binding profiles to enable switch management plugins (e.g., genericswitch) to configure the physical switch ports correctly.

Warning

Where the cache may be problematic is if you are re-cabling networker nodes on a fairly regular basis. While fundimentally such an action *is* a breaking change in itself in any operating environment, the cache will retain details for up to an hour and may also reflect incorrect details if the data sources (Ironic, or the YAML configuration) are not updated. Neutron guidance around changing configuration in such cases is also to change the agents which will reset the cache.

Prerequisites

Required Components

- **OpenStack Neutron** with ML2 plugin
- **OVN Backend** (Open Virtual Network) - required for ha_chassis_groups
- **L2VNI Mechanism Driver** - must be enabled and configured
- **Physical Network Switches** - configured for VLAN trunking
- **Switch Management Plugin** (e.g., genericswitch) - for VNIVLAN mapping
- **Network Nodes** - chassis that are members of OVN ha_chassis_groups

Network Architecture Requirements

Your deployment must use:

- OVN as the Neutron backend (ML2/OVN)
- HA chassis groups for gateway routers
- Network nodes with bridge-mappings to physical networks
- VLAN-capable physical switches connecting network nodes

Service Dependencies

The ironic-neutron-agent requires:

- Connectivity to Neutron API
- Connectivity to OVN Northbound database
- Connectivity to OVN Southbound database
- (Optional) Connectivity to Ironic API for enhanced switch information

Configuration

Enabling Trunk Reconciliation

Edit `/etc/neutron/neutron.conf` (or a separate config file in `/etc/neutron/neutron.conf.d/`) and add the `[l2vni]` section:

```
[l2vni]
# Enable L2VNI trunk reconciliation
enable_l2vni_trunk_reconciliation = True

# Enable event-driven L2VNI trunk reconciliation
enable_l2vni_trunk_reconciliation_events = True

# Baseline reconciliation interval (seconds)
l2vni_reconciliation_interval = 300

# Network node configuration file (fallback for switch info)
l2vni_network_nodes_config = /etc/neutron/l2vni_network_nodes.yaml

# Auto-create infrastructure networks
l2vni_auto_create_networks = True

# Subport anchor network name
l2vni_subport_anchor_network = l2vni-subport-anchor

# Network type for infrastructure networks (geneve or vxlan)
l2vni_subport_anchor_network_type = geneve

# Startup jitter to prevent thundering herd (seconds)
l2vni_startup_jitter_max = 60

# Ironic caching (if using Ironic for switch info)
ironic_cache_ttl = 3600
```

Configuration Options Reference

enable_l2vni_trunk_reconciliation

Type: Boolean

Default: `True`

Description: Enable periodic L2VNI trunk port reconciliation. When enabled, the agent runs reconciliation at regular intervals (`l2vni_reconciliation_interval`) to ensure eventual consistency.

This can be used independently or together with event-driven reconciliation. For production deployments, keeping this enabled is recommended as a safety net to catch missed events and handle agent restarts.

Set this to `False` to disable periodic reconciliation. Event-driven reconciliation (if enabled) will still work, but there will be no periodic safety net.

enable_l2vni_trunk_reconciliation_events

Type: Boolean

Default: `True`

Description: Enable OVN RowEvent-based event-driven L2VNI trunk reconciliation. When enabled, the agent watches OVN Northbound database for localnet port creation and deletion events and triggers immediate reconciliation. This eliminates the stale IDL cache issue and provides fast reconciliation response.

This can be used independently or together with periodic reconciliation. For production deployments, using both event-driven and periodic reconciliation is recommended. Event-driven provides immediate response, while periodic ensures eventual consistency.

Requires OVN IDL connection to be available. If the OVN IDL does not support event handlers, the agent will log a warning and fall back to periodic reconciliation only.

Set this to `False` to disable event-driven reconciliation. Periodic reconciliation (if enabled) will still work.

`l2vni_reconciliation_interval`
Type: Integer (seconds)

Default: `300` (5 minutes)

Minimum: `30`

Description: Baseline interval between reconciliation runs. This is the steady-state reconciliation frequency.

Tuning guidance:

- Smaller values (60-120s) provide faster convergence but increase API load
- Larger values (300-600s) reduce overhead but slow convergence
- For small deployments: 60-120 seconds
- For large deployments (100+ network nodes): 300-600 seconds

`l2vni_network_nodes_config`
Type: String (file path)

Default: `/etc/neutron/l2vni_network_nodes.yaml`

Description: Path to YAML configuration file containing network node trunk port configuration. This file serves as a fallback source for local_link_information when LLDP data is not available from OVN and Ironic.

See *Network Node Configuration File* for file format details.

`l2vni_auto_create_networks`
Type: Boolean

Default: `True`

Description: Automatically create Neutron networks for ha_chassis_groups and the subport anchor network if they do not exist.

When True (recommended):

- Networks are created automatically on first reconciliation
- Simplifies deployment and upgrades

- Networks are cleaned up when no longer needed

When False:

- Networks must be pre-created manually
- Network names must match expected patterns exactly
- Useful for environments with strict network creation policies

`l2vni_subport_anchor_network`

Type: String

Default: `l2vni-subport-anchor`

Description: Name of the shared network used for all trunk subports across all network nodes. This network is used to signal VLAN bindings to ML2 switch plugins and does not pass actual traffic.

All subports are created on this network regardless of which ha_chassis_group or overlay network they represent.

> **Note**
>
> If you change this value, ensure the network exists or set `l2vni_auto_create_networks = True`.

`l2vni_subport_anchor_network_type`

Type: String (enum)

Default: `geneve`

Choices: `geneve`, `vxlan`

Description: Network type to use for L2VNI infrastructure networks (both subport anchor and ha_chassis_group networks). These networks are used for metadata and modeling only, not for passing traffic.

Must match the overlay network type configured in your environment (ml2_type_drivers). If the specified type is not available, network creation will fail with an explicit error rather than falling back to an alternative type.

> **Warning**
>
> Ensure the selected network type is enabled in Neutrons ml2_type_drivers configuration before enabling trunk reconciliation.

`ironic_cache_ttl`

Type: Integer (seconds)

Default: `3600` (1 hour)

Minimum: `300` (5 minutes)

Description: Time-to-live for cached Ironic node and port data. Each system_id entry is cached independently and expires after this duration from when it was fetched.

A small amount of jitter (10-20%) is automatically added to spread cache refresh times across multiple agents, avoiding thundering herd issues when cache entries expire.

Tuning guidance:

- Smaller values (300-900s): More frequent Ironic API calls, faster detection of changes to node/port configurations
- Larger values (3600-7200s): Reduced API load, suitable for stable deployments where node configurations rarely change
- For deployments with frequent node changes: 300-600 seconds
- For stable deployments (1000+ nodes): 3600-7200 seconds

Performance impact:

In deployments with 1000 nodes Œ 8 ports each, efficient caching reduces per-reconciliation API calls from ~16,000 to ~2 (when cached).

`ironic_conductor_group`
Type: String

Default: `None` (no filtering)

Description: Ironic conductor group to filter nodes when querying for local_link_information data. This allows the agent to only query nodes managed by a specific conductor group, significantly reducing API load in large deployments with conductor group partitioning.

If not specified, all nodes are queried (subject to shard filtering).

Example use case:

In a deployment with separate conductor groups for different availability zones, set this to match the zone where trunk reconciliation is needed.

`ironic_shard`
Type: String

Default: `None` (no filtering)

Description: Ironic shard to filter nodes when querying for local_link_information data. This allows the agent to only query nodes in a specific shard, significantly reducing API load in large sharded deployments.

If not specified, all nodes are queried (subject to conductor group filtering).

Example use case:

In a deployment sharded by region (shard-us-west, shard-us-east), set this to match the region where trunk reconciliation is needed.

`l2vni_startup_jitter_max`
Type: Integer (seconds)

Default: `60`

Minimum: `0`

Description: Maximum random delay added to initial reconciliation start time after agent startup. This prevents thundering herd issues when multiple agents restart simultaneously (e.g., after a rolling upgrade).

Each agent will start reconciliation within 0 to `l2vni_startup_jitter_max` seconds of startup, spreading the initial load.

Recommended values:

- Single agent: `0` (no jitter needed)
- 2-5 agents: `30-60` seconds
- 6+ agents: `60-120` seconds

Network Node Configuration File

The network node configuration file provides fallback local_link_information when LLDP and Ironic data are not available.

File Format

The file is in YAML format at the path specified by `l2vni_network_nodes_config`:

```yaml
network_nodes:
  # Using system_id (explicit, no hostname lookup needed)
  - system_id: 0f563ca5-4a94-4d26-a21e-a4ce3dbcd372
    trunks:
      - physical_network: physnet1
        local_link_information:
          - switch_id: "00:11:22:33:44:55"
            port_id: "Ethernet1/1"
            switch_info: "tor-switch-1"
      - physical_network: physnet2
        local_link_information:
          - switch_id: "aa:bb:cc:dd:ee:ff"
            port_id: "GigabitEthernet1/0/1"
            switch_info: "tor-switch-2"

  # Using hostname with LAG/bonding (multiple links)
  - hostname: network-node-2.example.com
    trunks:
      - physical_network: physnet1
        local_link_information:
          - switch_id: "22:57:f8:dd:03:01"
            port_id: "Ethernet1/3"
            switch_info: "leaf01.netlab.example.com"
          - switch_id: "22:57:f8:dd:03:01"
            port_id: "Ethernet1/5"
            switch_info: "leaf01.netlab.example.com"
```

Field Descriptions

network_nodes (required)
: List of network node configurations

system_id or hostname (one required)
: Network nodes can be identified by either:

- **system_id**: The OVN chassis UUID (chassis.name in OVN Southbound). Explicit and requires no lookup. Takes priority if both are specified.
- **hostname**: The OVN chassis hostname (chassis.hostname in OVN Southbound). Human-readable and survives chassis reinstalls, but requires an OVN SB lookup to resolve to the chassis UUID.

You can find both values with:

```
ovn-sbctl --columns name,hostname list Chassis
```

Choose based on your needs: `system_id` for explicitness and performance, `hostname` for maintainability and readability.

trunks (required)
List of trunk configurations for this network node

physical_network (required)
The physical network name (must match `network_vlan_ranges` and `ovn-bridge-mappings` configuration)

local_link_information (required)
List of switch connection information dictionaries. Each entry contains:

- **switch_id**: Switch MAC address or identifier
- **port_id**: Switch port name/identifier
- **switch_info**: Optional switch hostname or description

Specify a single-item list for single links, or multiple items for LAG/bonding configurations. All links are aggregated into the anchor ports `binding:profile['local_link_information']` list and passed to switch management plugins to enable automatic switch port configuration.

Multi-Agent Deployment

Hash Ring Distribution

When multiple ironic-neutron-agents are deployed, they use a hash ring to distribute work. Each chassis is hashed to a specific agent, and only that agent manages trunks for that chassis.

Benefits:

- Load distribution across agents
- Reduced API call volume
- Parallel processing of reconciliation tasks

How It Works:

1. Agents register with Tooz coordinator (typically backed by etcd or Redis)
2. A consistent hash ring is built from agent memberships
3. Each chassis system-id is hashed to determine the owning agent
4. During reconciliation, agents skip chassis not in their hash ring segment

Example: 3 agents managing 10 chassis:

- Agent A manages: chassis-1, chassis-4, chassis-7, chassis-10

- Agent B manages: chassis-2, chassis-5, chassis-8
- Agent C manages: chassis-3, chassis-6, chassis-9

Cleanup Considerations

Important: Cleanup operations do **not** use hash ring filtering.

When orphaned trunks are detected (chassis removed from ha_chassis_group or deleted entirely), all agents will attempt cleanup. This is intentional:

Scenario: Agent A managed chassis-5, then Agent A crashes. Chassis-5 is deleted from OVN. Agents B and C both detect the orphaned trunk and attempt cleanup. The first agent to run cleanup succeeds; the other gets a not found error (harmless).

This approach provides:

- **Resilience**: Cleanup happens even if the original managing agent is down
- **Simplicity**: No need to track previous ownership
- **Correctness**: No orphaned resources due to agent failures

The cost is minimal: redundant API calls that return 404, logged as warnings.

High Availability

For production deployments, run at least 3 ironic-neutron-agents:

```
# On controller-1
systemctl start ironic-neutron-agent

# On controller-2
systemctl start ironic-neutron-agent

# On controller-3
systemctl start ironic-neutron-agent
```

If an agent fails:

1. Tooz coordinator detects the failure
2. Hash ring is recalculated
3. Remaining agents automatically take over the failed agents chassis
4. Reconciliation continues without interruption

Deployment Guide

Step 1: Enable the L2VNI Mechanism Driver

Ensure the L2VNI mechanism driver is enabled and configured. See *L2VNI Mechanism Driver Configuration* for details.

Step 2: Configure Trunk Reconciliation

Edit `/etc/neutron/neutron.conf`:

```
[l2vni]
enable_l2vni_trunk_reconciliation = True
enable_l2vni_trunk_reconciliation_events = True
l2vni_reconciliation_interval = 300
l2vni_auto_create_networks = True
l2vni_subport_anchor_network_type = geneve
l2vni_startup_jitter_max = 60
```

Step 3: (Optional) Create Network Node Config

If LLDP data is not available in OVN, create `/etc/neutron/l2vni_network_nodes.yaml`:

```
network_nodes:
  - hostname: network-node-1.example.com
    trunks:
      - physical_network: physnet1
        local_link_information:
          - switch_id: "00:11:22:33:44:55"
            port_id: "Ethernet1/1"
            switch_info: "tor-switch-1"
```

Note

See *Network Node Configuration File* for complete format details, including multi-link LAG/bonding configurations.

Step 4: Restart ironic-neutron-agent

```
systemctl restart ironic-neutron-agent
```

Step 5: Verify Operation

Check logs for successful reconciliation:

```
journalctl -u ironic-neutron-agent -f | grep -E "L2VNI|OVN event"
```

You should see:

```
Registered OVN event handler for L2VNI localnet port creation
Started L2VNI trunk reconciliation loop (interval: 300s, initial delay: 345s␣
↪with 45s jitter)
Starting L2VNI trunk reconciliation
Discovered trunk l2vni-trunk-network-node-1-physnet1
Added subport port-uuid (VLAN 100) to trunk trunk-uuid
L2VNI trunk reconciliation completed successfully
```

When a baremetal port is bound and a localnet port is created, you should see event-driven reconciliation:

```
OVN localnet port CREATE event: neutron-abc123-localnet-physnet1 (network:␣
↪abc123, owned: True)
Triggering L2VNI trunk reconciliation due to localnet port CREATE event
Starting L2VNI trunk reconciliation
L2VNI trunk reconciliation completed successfully
```

Step 6: Create Test Overlay Network

Create a test setup to verify trunk reconciliation:

```
# Create overlay network
openstack network create test-overlay

# Create subnet
openstack subnet create --network test-overlay \
  --subnet-range 192.168.100.0/24 test-subnet

# Create router with external gateway
openstack router create test-router
openstack router set --external-gateway public test-router

# Add overlay network to router
openstack router add subnet test-router test-subnet

# Create baremetal port
openstack port create \
  --network test-overlay \
  --vnic-type baremetal \
  --binding-profile physical_network=physnet1 \
  test-bm-port
```

After the next reconciliation cycle, verify trunk subports:

```
openstack network trunk list
openstack network trunk show <trunk-id>
```

You should see a subport with the VLAN allocated for test-overlay.

Monitoring and Operations

Log Messages

Normal Operation:

```
Started L2VNI trunk reconciliation loop
Starting L2VNI trunk reconciliation
Discovered trunk l2vni-trunk-system-1-physnet1
Added subport port-123 (VLAN 100) to trunk trunk-456
L2VNI trunk reconciliation completed successfully
```

OVN Event-Driven Reconciliation:

```
Registered OVN event handler for L2VNI localnet port creation
OVN localnet port CREATE event: neutron-abc123-localnet-physnet1 (network:␣
↪abc123, owned: True)
Triggering L2VNI trunk reconciliation due to localnet port CREATE event
Starting L2VNI trunk reconciliation
L2VNI trunk reconciliation completed successfully
OVN localnet port DELETE event: neutron-abc123-localnet-physnet1 (network:␣
↪abc123, owned: True)
Triggering L2VNI trunk reconciliation due to localnet port DELETE event
```

Event-Driven Fast Mode:

```
Router notification received, triggering fast reconciliation
Switched to fast reconciliation mode (interval: 90s, duration: 600s)
Exiting fast reconciliation mode, returning to baseline interval
```

Cleanup Operations:

```
Cleaning up orphaned trunk trunk-789 for chassis deleted-system physnet1
Deleted orphaned trunk trunk-789
Deleted orphaned anchor port port-999
Cleaning up orphaned ha_chassis_group network net-111 for group deleted-group
```

Warnings (Expected During Cleanup):

```
Failed to delete subport subport-222
Failed to delete trunk trunk-333
```

These warnings are normal when multiple agents attempt cleanup simultaneously. The first agent succeeds; others log warnings.

Metrics and Health Indicators

Monitor these indicators for healthy operation:

1. **Reconciliation Completion**: Should see completed successfully messages at configured intervals
2. **API Error Rate**: Occasional 404 errors during cleanup are normal; frequent 500 errors indicate problems
3. **Reconciliation Duration**: Should complete in seconds; if taking minutes, check for API performance issues
4. **Orphaned Resource Count**: Should be zero or near-zero in steady state

Troubleshooting

Reconciliation Not Running

Symptom: No L2VNI reconciliation log messages.

Possible Causes:

1. **Feature disabled**: `enable_l2vni_trunk_reconciliation = False`

   Solution: Set to `True` in config and restart agent.

2. **OVN connection failure**: Agent cannot connect to OVN NB/SB databases.

   Solution: Check OVN connection settings in config. Verify OVN services are running and accessible.

3. **No ha_chassis_groups**: No network nodes are members of ha_chassis_groups.

   Solution: This is normal if you have no routers with gateway ports. Create a router with external gateway to trigger ha_chassis_group creation.

Trunks Not Created

Symptom: Reconciliation runs but no trunks appear.

Possible Causes:

1. **No chassis in ha_chassis_groups**: Chassis exists in OVN but not assigned to any ha_chassis_group.

   Solution: Create a router with external gateway. OVN will automatically assign gateway chassis.

2. **Missing bridge-mappings**: Chassis lacks `ovn-bridge-mappings` in external_ids.

   Solution: Configure bridge-mappings on the chassis:

   ```
   ovs-vsctl set Open_vSwitch . \
     external-ids:ovn-bridge-mappings=physnet1:br-provider
   ```

3. **Network creation disabled**: `l2vni_auto_create_networks = False` but ha_chassis_group network doesnt exist.

   Solution: Either enable auto-creation or manually create network:

   ```
   openstack network create l2vni-ha-group-{group-uuid}
   ```

Subports Not Added

Symptom: Trunks exist but have no subports.

Possible Causes:

1. **No overlay networks**: No VXLAN/Geneve networks are attached to routers.

   Solution: This is normal. Create overlay networks and attach to routers.

2. **No VLAN segments allocated**: L2VNI mechanism driver didnt allocate VLAN segments.

   Solution: Check that baremetal ports exist on overlay networks. Verify L2VNI mechanism driver is enabled and working.

3. **Subport anchor network missing**: Subport creation fails because anchor network doesnt exist.

   Solution: Enable auto-creation or manually create:

   ```
   openstack network create l2vni-subport-anchor
   ```

Missing Switch Information

Symptom: Ports created but binding profile lacks local_link_information.

Possible Causes:

1. **No LLDP data in OVN**: OVN doesnt have LLDP information for the chassis.

   Solution: Ensure switches are sending LLDP and OVN is configured to collect it, or provide fallback config file.

2. **Ironic not available**: Agent cannot query Ironic API.

   Solution: Check Ironic connectivity or provide fallback config file.

3. **Config file missing/incorrect**: Fallback config file doesnt exist or has wrong format.

   Solution: Create config file following the format in *Network Node Configuration File*.

Reconciliation Taking Too Long

Symptom: Reconciliation runs for minutes instead of seconds.

Possible Causes:

1. **Large number of network nodes**: Many chassis with many physical networks.

   Solution: Increase `l2vni_reconciliation_interval` to reduce frequency. Consider deploying more agents for load distribution.

2. **Neutron API slow**: API calls taking long time.

   Solution: Investigate Neutron API performance. Check database load.

3. **OVN database queries slow**: Queries to OVN NB/SB taking long time.

   Solution: Check OVN database performance. Ensure indexes are healthy.

4. **Ironic queries slow**: Querying thousands of Ironic nodes/ports per run.

   Solution: Enable Ironic caching and use conductor_group/shard filtering:

   ```
   [l2vni]
   ironic_cache_ttl = 3600
   ironic_conductor_group = network-nodes
   ironic_shard = region-1
   ```

Infrastructure Network Creation Fails

Symptom: Reconciliation fails with errors about network creation.

Example Error:

```
ERROR Failed to create L2VNI network 'l2vni-subport-anchor' with type
'geneve'. This indicates a misconfiguration - the requested network type
is not available in your environment.
```

Possible Causes:

1. **Network type not enabled**: Configured network type (geneve or vxlan) is not enabled in Neutrons ml2_type_drivers.

   Solution: Add the network type to ml2_type_drivers in neutron.conf:

   ```
   [ml2]
   type_drivers = flat,vlan,geneve,vxlan
   ```

   Then restart neutron-server.

2. **Wrong network type configured**: `l2vni_subport_anchor_network_type` doesnt match your deployments overlay network type.

   Solution: Set to match your environment:

   ```
   [l2vni]
   # For VXLAN-based deployments
   l2vni_subport_anchor_network_type = vxlan

   # For Geneve-based deployments (default)
   l2vni_subport_anchor_network_type = geneve
   ```

   Then restart ironic-neutron-agent.

Agent Crashes During Reconciliation

Symptom: ironic-neutron-agent crashes or restarts during reconciliation.

Check Logs:

```
journalctl -u ironic-neutron-agent --since "1 hour ago"
```

Possible Causes:

1. **Uncaught exception**: Bug in reconciliation code.

   Solution: Report the bug with full traceback. As a workaround, disable trunk reconciliation until fixed.

2. **Memory exhaustion**: Agent runs out of memory in large deployments.

   Solution: Increase agent memory limits. Consider deploying more agents to distribute load.

3. **Deadlock or timeout**: Operation hangs waiting for response.

   Solution: Check network connectivity to Neutron/OVN. Review timeout settings.

OVN Event Reconciliation Not Working

Symptom: No event-driven reconciliation logs, only periodic reconciliation.

Check Logs:

```
journalctl -u ironic-neutron-agent | grep "OVN event handler"
```

Possible Causes:

1. **Feature disabled**: `enable_l2vni_trunk_reconciliation_events = False`

   Solution: Set to `True` in config and restart agent.

2. **OVN IDL not available**: Agent cannot connect to OVN Northbound database.

   Solution: Check OVN connection settings. Verify OVN services are running and accessible. Check for error messages about OVN connection failures.

3. **Event handler registration failed**: Agent logged warning about IDL not supporting event handlers.

   Solution: This indicates the OVN IDL connection was not established at agent startup. Check OVN connectivity and restart the agent. If the problem persists, check for OVN version compatibility issues.

Expected Behavior:

When working correctly, you should see this log message at agent startup:

```
Registered OVN event handler for L2VNI localnet port creation
```

If you see this warning instead, event-driven reconciliation is not active:

```
OVN IDL does not support event handlers, falling back to periodic␣
↪reconciliation only
```

Upgrade Considerations

Upgrading from Previous Versions

When upgrading to a version with trunk reconciliation:

1. **First upgrade**: Feature is disabled by default. No impact.
2. **Enabling the feature**:
   - Add `[l2vni]` configuration to neutron.conf
   - Set `enable_l2vni_trunk_reconciliation = True`
   - Restart ironic-neutron-agent
3. **First reconciliation**: Agent will:
   - Create infrastructure networks
   - Create anchor ports and trunks for existing network nodes
   - Add subports for existing overlay networks

   This initial reconciliation may take longer than usual. Watch logs for progress.

Rolling Upgrades

The startup jitter feature (`l2vni_startup_jitter_max`) is specifically designed to handle rolling upgrades gracefully:

Scenario: 3 agents running, performing rolling upgrade:

1. Stop agent-1, upgrade, restart starts reconciliation after random delay (0-60s)

2. Stop agent-2, upgrade, restart starts reconciliation after different delay (0-60s)
3. Stop agent-3, upgrade, restart starts reconciliation after different delay (0-60s)

This prevents all agents from hitting Neutron API simultaneously after restart.

Best Practices:

- Use default `l2vni_startup_jitter_max = 60` or higher
- Upgrade one agent at a time
- Wait for agent to complete first reconciliation before upgrading next
- Monitor API load during upgrade

Compatibility

Required OpenStack Release:

- Queens or later (requires OVN backend and ML2/OVN)

L2VNI Mechanism Driver:

- Must be version 7.1.0 or later for full compatibility
- Trunk reconciliation works independently but is designed to complement the mechanism driver

Ironic Integration:

- Optional: Ironic API is used for enhanced switch information if available
- Works without Ironic using LLDP or config file fallback

Disabling the Feature

The agent supports flexible configuration of reconciliation mechanisms:

To disable event-driven L2VNI trunk reconciliation only (keep periodic):

1. Set `enable_l2vni_trunk_reconciliation_events = False` in config
2. Restart ironic-neutron-agent
3. Event-driven reconciliation stops; periodic reconciliation continues

To disable periodic reconciliation only (keep event-driven):

1. Set `enable_l2vni_trunk_reconciliation = False` in config
2. Restart ironic-neutron-agent
3. Periodic reconciliation stops; event-driven reconciliation continues
4. Note: Without periodic reconciliation, there is no safety net for missed events

To disable trunk reconciliation completely:

1. Set both `enable_l2vni_trunk_reconciliation = False` and `enable_l2vni_trunk_reconciliation_events = False` in config
2. Restart ironic-neutron-agent
3. All reconciliation stops; existing trunks remain unchanged

To fully clean up:

```
# List L2VNI trunks
openstack network trunk list | grep l2vni-trunk

# Delete each trunk (subports are automatically removed)
openstack network trunk delete <trunk-id>

# Delete infrastructure networks
openstack network list | grep l2vni-
openstack network delete <network-id>
```

Performance Tuning

Small Deployments (< 10 network nodes)

Optimize for fast convergence:

```
[l2vni]
l2vni_reconciliation_interval = 60
l2vni_startup_jitter_max = 10
```

Medium Deployments (10-50 network nodes)

Balance convergence and overhead:

```
[l2vni]
l2vni_reconciliation_interval = 180
l2vni_startup_jitter_max = 30
```

Large Deployments (50+ network nodes)

Optimize for reduced API load:

```
[l2vni]
l2vni_reconciliation_interval = 600
l2vni_startup_jitter_max = 120
```

Ironic Integration Performance

For deployments using Ironic for switch connection information:

Small Ironic Deployments (< 100 nodes)

Fast cache refresh:

```
[l2vni]
ironic_cache_ttl = 600
```

Medium Ironic Deployments (100-500 nodes)

Balanced refresh:

```
[l2vni]
ironic_cache_ttl = 1800
```

Large Ironic Deployments (500+ nodes)

Optimize for API load reduction:

```
[l2vni]
ironic_cache_ttl = 7200
# Optional: Filter to reduce query scope
ironic_conductor_group = network-nodes
ironic_shard = region-west
```

Sharded Deployments

Use shard filtering to query only relevant nodes:

```
[l2vni]
ironic_shard = shard-region-1
ironic_cache_ttl = 3600
```

See Also

- *Router HA Binding for VLAN Networks* - Router HA Binding for VLAN Networks
- *L2VNI Mechanism Driver Configuration* - L2VNI Mechanism Driver
- *Configuration Reference* - Agent Configuration
- *Contributing* - Contributing Guide
- OpenStack Neutron Documentation: https://docs.openstack.org/neutron/
- OVN Documentation: https://www.ovn.org/

CHAPTER

SIX

CONTRIBUTOR GUIDE

6.1 Contributing

This document provides some necessary points for developers to consider when writing and reviewing networking-baremetal code.

6.1.1 Getting Started

If youre completely new to OpenStack and want to contribute to the networking-baremetal project, please start by familiarizing yourself with the Infra Teams Developer Guide. This will help you get your accounts set up in Launchpad and Gerrit, familiarize you with the workflow for the OpenStack continuous integration and testing systems, and help you with your first commit.

LaunchPad Project

Most of the tools used for OpenStack require a launchpad.net ID for authentication.

See also

- https://launchpad.net
- https://launchpad.net/ironic

Related Projects

Networking Baremetal is tightly integrated with the ironic and neutron projects. Ironic and its related projects are developed by the same community.

See also

- https://launchpad.net/ironic
- https://launchpad.net/neutron

Project Hosting Details

Bug tracker
https://bugs.launchpad.net/networking-baremetal

Mailing list (prefix Subject line with `[ironic][networking-baremetal]`)
http://lists.openstack.org/cgi-bin/mailman/listinfo/openstack-discuss

Code Hosting
https://opendev.org/openstack/networking-baremetal

Code Review
https://review.opendev.org/#/q/status:open+project:openstack/networking-baremetal,n,z

6.1.2 Developer quick-starts

These are quick walk throughs to get you started developing code for networking-baremetal. These assume you are already familiar with submitting code reviews to an OpenStack project.

Deploying networking-baremetal with DevStack

DevStack may be configured to deploy networking-baremetal Networking service plugin. It is highly recommended to deploy on an expendable virtual machine and not on your personal work station. Deploying networking-baremetal with DevStack requires a machine running Ubuntu 14.04 (or later) or Fedora 20 (or later).

See also

http://docs.openstack.org/devstack/latest

Create `devstack/local.conf` with minimal settings required to enable networking-baremetal with ironic. Here is an example of local.conf:

```
cd devstack
cat >local.conf <<END
[[local|localrc]]
# Credentials
ADMIN_PASSWORD=password
DATABASE_PASSWORD=password
RABBIT_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=password
SWIFT_HASH=password
SWIFT_TEMPURL_KEY=password

# Enable networking-baremetal plugin
enable_plugin networking-baremetal https://opendev.org/openstack/networking-
↪baremetal.git
enable_service ir-neutronagt

# Enable ironic plugin
enable_plugin ironic https://opendev.org/openstack/ironic
enable_service networking_baremetal

# Enable neutron which is required by ironic and disable nova-network.
disable_service n-net
disable_service n-novnc
enable_service q-svc
enable_service q-agt
```

(continues on next page)

(continued from previous page)

```
enable_service q-dhcp
enable_service q-l3
enable_service q-meta
enable_service neutron

# Enable swift for agent_* drivers
enable_service s-proxy
enable_service s-object
enable_service s-container
enable_service s-account

# Disable horizon
disable_service horizon

# Disable heat
disable_service heat h-api h-api-cfn h-api-cw h-eng

# Disable cinder
disable_service cinder c-sch c-api c-vol

# Swift temp URL's are required for agent_* drivers.
SWIFT_ENABLE_TEMPURLS=True

# Create 3 virtual machines to pose as ironic's baremetal nodes.
IRONIC_VM_COUNT=3
IRONIC_VM_SSH_PORT=22
IRONIC_BAREMETAL_BASIC_OPS=True
DEFAULT_INSTANCE_TYPE=baremetal

# Enable additional hardware types, if needed.
#IRONIC_ENABLED_HARDWARE_TYPES=ipmi,fake-hardware
# Don't forget that many hardware types require enabling of additional
# interfaces, most often power and management:
#IRONIC_ENABLED_MANAGEMENT_INTERFACES=ipmitool,fake
#IRONIC_ENABLED_POWER_INTERFACES=ipmitool,fake
# The 'ipmi' hardware type's default deploy interface is 'iscsi'.
# This would change the default to 'direct':
#IRONIC_DEFAULT_DEPLOY_INTERFACE=direct

# Change this to alter the default driver for nodes created by devstack.
# This driver should be in the enabled list above.
IRONIC_DEPLOY_DRIVER=ipmi

# The parameters below represent the minimum possible values to create
# functional nodes.
IRONIC_VM_SPECS_RAM=1280
IRONIC_VM_SPECS_DISK=10

# Size of the ephemeral partition in GB. Use 0 for no ephemeral partition.
```

(continues on next page)

(continued from previous page)

```
IRONIC_VM_EPHEMERAL_DISK=0

# To build your own IPA ramdisk from source, set this to True
IRONIC_BUILD_DEPLOY_RAMDISK=False

VIRT_DRIVER=ironic

# By default, DevStack creates a 10.0.0.0/24 network for instances.
# If this overlaps with the hosts network, you may adjust with the
# following.
NETWORK_GATEWAY=10.1.0.1
FIXED_RANGE=10.1.0.0/24
FIXED_NETWORK_SIZE=256

# Log all output to files
LOGFILE=$HOME/devstack.log
LOGDIR=$HOME/logs
IRONIC_VM_LOG_DIR=$HOME/ironic-bm-logs

END
```

Deploying networking-baremetal and multi-tenant networking with DevStack

DevStack may be configured to deploy networking-baremetal Networking service plugin together with networking-generic-switch for multi-tenant networking. It is highly recommended to deploy on an expendable virtual machine and not on your personal work station. Deploying networking-baremetal with DevStack requires a machine running Ubuntu 14.04 (or later) or Fedora 20 (or later).

See also

http://docs.openstack.org/devstack/latest

Create `devstack/local.conf` with minimal settings required to enable networking-baremetal with ironic and networking-generic-switch for multi-tenant networking. Here is an example of local.conf:

```
[[local|localrc]]

# Credentials
ADMIN_PASSWORD=password
DATABASE_PASSWORD=password
RABBIT_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=password
SWIFT_HASH=password
SWIFT_TEMPURL_KEY=password

# Install networking-generic-switch Neutron ML2 driver that interacts with OVS
enable_plugin networking-generic-switch https://opendev.org/openstack/
↪networking-generic-switch
```

(continues on next page)

(continued from previous page)

```
# Enable networking-baremetal plugin
enable_plugin networking-baremetal https://opendev.org/openstack/networking-
↪baremetal.git
enable_service networking_baremetal
enable_service ir-neutronagt

# Add link local info when registering Ironic node
IRONIC_USE_LINK_LOCAL=True

IRONIC_ENABLED_NETWORK_INTERFACES=flat,neutron
IRONIC_NETWORK_INTERFACE=neutron

#Networking configuration
OVS_PHYSICAL_BRIDGE=brbm
PHYSICAL_NETWORK=mynetwork
IRONIC_PROVISION_NETWORK_NAME=ironic-provision
IRONIC_PROVISION_PROVIDER_NETWORK_TYPE=vlan
IRONIC_PROVISION_SUBNET_PREFIX=10.0.5.0/24
IRONIC_PROVISION_SUBNET_GATEWAY=10.0.5.1
Q_PLUGIN=ml2
ENABLE_TENANT_VLANS=True
Q_ML2_TENANT_NETWORK_TYPE=vlan
TENANT_VLAN_RANGE=100:150
Q_USE_PROVIDERNET_FOR_PUBLIC=False

# Enable segments service_plugin for routed networks
Q_SERVICE_PLUGIN_CLASSES=neutron.services.l3_router.l3_router_plugin.
↪L3RouterPlugin,segments
IRONIC_USE_NEUTRON_SEGMENTS=True

# Configure ironic from ironic devstack plugin.
enable_plugin ironic https://opendev.org/openstack/ironic

# Enable Ironic API and Ironic Conductor
enable_service ironic
enable_service ir-api
enable_service ir-cond

# Enable Neutron which is required by Ironic and disable nova-network.
disable_service n-net
disable_service n-novnc
enable_service q-svc
enable_service q-agt
enable_service q-dhcp
enable_service q-l3
enable_service q-meta
enable_service neutron
```

(continues on next page)

(continued from previous page)

```
# Enable Swift for agent_* drivers
enable_service s-proxy
enable_service s-object
enable_service s-container
enable_service s-account

# Disable Horizon
disable_service horizon

# Disable Heat
disable_service heat h-api h-api-cfn h-api-cw h-eng

# Disable Cinder
disable_service cinder c-sch c-api c-vol

# Swift temp URL's are required for agent_* drivers.
SWIFT_ENABLE_TEMPURLS=True

# Create 3 virtual machines to pose as Ironic's baremetal nodes.
IRONIC_VM_COUNT=3
IRONIC_BAREMETAL_BASIC_OPS=True
DEFAULT_INSTANCE_TYPE=baremetal

# Enable additional hardware types, if needed.
#IRONIC_ENABLED_HARDWARE_TYPES=ipmi,fake-hardware
# Don't forget that many hardware types require enabling of additional
# interfaces, most often power and management:
#IRONIC_ENABLED_MANAGEMENT_INTERFACES=ipmitool,fake
#IRONIC_ENABLED_POWER_INTERFACES=ipmitool,fake
# The 'ipmi' hardware type's default deploy interface is 'iscsi'.
# This would change the default to 'direct':
#IRONIC_DEFAULT_DEPLOY_INTERFACE=direct

# Change this to alter the default driver for nodes created by devstack.
# This driver should be in the enabled list above.
IRONIC_DEPLOY_DRIVER=ipmi

# The parameters below represent the minimum possible values to create
# functional nodes.
IRONIC_VM_SPECS_RAM=1024
IRONIC_VM_SPECS_DISK=10

# Size of the ephemeral partition in GB. Use 0 for no ephemeral partition.
IRONIC_VM_EPHEMERAL_DISK=0

# To build your own IPA ramdisk from source, set this to True
IRONIC_BUILD_DEPLOY_RAMDISK=False

VIRT_DRIVER=ironic
```

(continues on next page)

(continued from previous page)

```
# By default, DevStack creates a 10.0.0.0/24 network for instances.
# If this overlaps with the hosts network, you may adjust with the
# following.
NETWORK_GATEWAY=10.1.0.1
FIXED_RANGE=10.1.0.0/24
FIXED_NETWORK_SIZE=256

# Log all output to files
LOGFILE=$HOME/devstack.log
LOGDIR=$HOME/logs
IRONIC_VM_LOG_DIR=$HOME/ironic-bm-logs
```

Virtual lab with virtual switch and netconf-openconfig Device Driver

Ansible playbooks that can be used to set up a lab for developing networking-baremetal network device integration is hosted on GitHub.

6.1.3 Full networking-baremetal python API reference

- modindex

networking_baremetal

networking_baremetal package

Subpackages

networking_baremetal.agent package

Submodules

networking_baremetal.agent.agent_config module

Configuration options for ironic-neutron-agent.

networking_baremetal.agent.agent_config.**list_opts**()

Return a list of oslo_config options for config generation.

Returns
list of (group_name, options) tuples

networking_baremetal.agent.agent_config.**register_agent_opts**(*conf*)

Register all agent configuration options.

Parameters
conf oslo_config.cfg.ConfigOpts instance

networking_baremetal.agent.agent_config.**register_baremetal_agent_opts**(*conf*)

Register baremetal agent configuration options (deprecated).

Prefer register_agent_opts() instead.

Parameters
conf oslo_config.cfg.ConfigOpts instance

networking_baremetal.agent.agent_config.**register_l2vni_opts**(*conf*)

Register L2VNI configuration options (deprecated).

Prefer register_agent_opts() instead.

Parameters
conf oslo_config.cfg.ConfigOpts instance

networking_baremetal.agent.ironic_neutron_agent module

class networking_baremetal.agent.ironic_neutron_agent.**BaremetalNeutronAgent**

Bases: ServiceBase

cleanup_stale_agents()

Cleans up stale baremetal agents

This method identifies baremetal agents that are marked as inactive in the Neutron server and are not associated with any nodes in Ironic. It then deletes these stale agents.

get_template_node_state(*node_uuid*)

reset()

Reset service.

Called in case a service running in daemon mode receives SIGHUP.

start()

Start service.

stop(*failure=False*)

Stop service.

wait()

Wait for service to complete.

class networking_baremetal.agent.ironic_neutron_agent.**HashRingMemberManagerNotificationEndpoint**

Bases: object

Class variables members and hashring is shared by all instances

filter_rule = <oslo_messaging.notify.filter.NotificationFilter object>

hashring = <tooz.hashring.HashRing object>

info(*ctxt*, *publisher_id*, *event_type*, *payload*, *metadata*)

members = []

networking_baremetal.agent.ironic_neutron_agent.**list_opts**()

networking_baremetal.agent.ironic_neutron_agent.**main**()

networking_baremetal.agent.l2vni_trunk_manager module

L2VNI Trunk Reconciliation Manager.

Manages trunk ports and subports for OVN network nodes to ensure only required VLANs are trunked to each chassis based on ha_chassis_group membership and network requirements.

Architecture: - One Neutron network per OVN ha_chassis_group for anchor port modeling - One shared subport anchor network for all trunk subports - Anchor ports (trunk parents) attach to ha_chassis_group networks - Subports signal VLAN bindings to ML2 switch plugins - Stateless reconciliation based on current OVN/Neutron state

class networking_baremetal.agent.l2vni_trunk_manager.**L2VNITrunkManager**(*neutron_client*, *ovn_nb_idl*, *ovn_sb_idl*, *ironic_client*, *member_manager=None*, *agent_id=None*)

Bases: object

Manages L2VNI trunk ports and subports for network nodes.

reconcile()

Main reconciliation entry point.

Performs stateless reconciliation of trunk infrastructure: 1. Ensure ha_chassis_group networks exist 2. Ensure subport anchor network exists 3. Discover/create trunk ports for chassis 4. Calculate required VLANs per chassis from OVN state 5. Reconcile subports to match requirements 6. Clean up unused infrastructure

reconcile_single_vlan(*network_id*, *physnet*, *vlan_id*, *action='add'*)

Targeted reconciliation for a single VLAN.

Called by OVN event handlers when a specific localnet port is created or deleted. Much faster than full reconciliation because we already know which VLAN to add/remove.

Parameters

- **network_id** Neutron network UUID (tenant network with VLAN)
- **physnet** Physical network name
- **vlan_id** VLAN ID to add or remove
- **action** add for CREATE events, remove for DELETE events

networking_baremetal.agent.ovn_client module

OVN client connections for agent.

class networking_baremetal.agent.ovn_client.**AgentOvnNbIdl**(*remote*, *schema*, ***kwargs*)

Bases: OvsdbIdl

Custom OVN NB IDL with event handler support for agent.

Extends the standard OvsdbIdl to add RowEvent notification support. This allows the agent to watch for specific OVN database changes (e.g., localnet port creation) and trigger immediate reconciliation.

notify(*event*, *row*, *updates=None*)

Called by IDL when database changes occur.

Forwards notifications to registered RowEvent handlers.

class networking_baremetal.agent.ovn_client.**AgentOvnSbIdl**(*remote*, *schema*, ***kwargs*)

Bases: OvsdbIdl

Custom OVN SB IDL with event handler support for agent.

Extends the standard OvsdbIdl to add RowEvent notification support. This allows the agent to watch for specific OVN database changes and trigger immediate reconciliation if needed.

notify(*event*, *row*, *updates=None*)

Called by IDL when database changes occur.

Forwards notifications to registered RowEvent handlers.

networking_baremetal.agent.ovn_client.**get_ovn_nb_event_idl**()

Get OVN Northbound IDL connection for event watching only.

This connection registers only the minimal set of tables needed for event watching, significantly reducing event notification overhead. Use this connection for registering RowEvent handlers.

For queries and updates, use get_ovn_nb_idl() instead.

Returns
OVN NB API instance (event-watching connection)

networking_baremetal.agent.ovn_client.**get_ovn_nb_idl**()

Get OVN Northbound IDL connection.

Returns
OVN NB API instance

networking_baremetal.agent.ovn_client.**get_ovn_sb_idl**()

Get OVN Southbound IDL connection.

Returns
OVN SB API instance

networking_baremetal.agent.ovn_events module

OVN RowEvent handlers for L2VNI reconciliation.

class networking_baremetal.agent.ovn_events.**HAChassisGroupNetworkEvent**(*agent*)

Bases: BaseEvent

Trigger router HA binding for HA Chassis group create/update.

Watches for CREATE and UPDATE events on HA_Chassis_Group table where the group is network-level (has neutron:network_id in external_ids but not neutron:router_id).

Uses hash ring to filter events so only the agent responsible for the network processes the event.

Fixes LP#2144458 by enabling immediate router interface binding instead of waiting for periodic reconciliation. Related to LP#1995078 (networking-baremetal side of the solution).

events = ('create', 'update')

match_fn(*event*, *row*, *old=None*)

Filter for HA chassis groups with network_id owned by this agent.

Returns True only if: 1. HA chassis group has neutron:network_id in external_ids 2. This agent owns the network (hash ring check)

Note: We process groups with network_id regardless of whether they also have router_id. In unified HA chassis group scenarios, the same group is used for both the network and router.

Parameters

- **event** Event type (ROW_CREATE or ROW_UPDATE)
- **row** OVN HA_Chassis_Group row
- **old** Previous row state (for UPDATE events)

Returns

True if event should be processed, False otherwise

run(*event*, *row*, *old*)

Trigger router interface binding for the network.

Parameters

- **event** Event type (ROW_CREATE or ROW_UPDATE)
- **row** OVN HA_Chassis_Group row
- **old** Previous row state (for UPDATE events)

table: str = 'HA_Chassis_Group'

class networking_baremetal.agent.ovn_events.**LocalnetPortEvent**(*agent*)

Bases: BaseEvent

Trigger L2VNI reconciliation when localnet ports are created or deleted.

Watches for CREATE and DELETE events on Logical_Switch_Port table where type=localnet and the port name follows L2VNI naming convention (contains -localnet-).

Uses hash ring to filter events so only the agent responsible for the network processes the event.

CREATE events trigger immediate reconciliation to add required subports. DELETE events trigger immediate reconciliation to remove obsolete subports, ensuring fast cleanup for security and resource isolation.

events = ('create', 'delete')

match_fn(*event*, *row*, *old=None*)

Filter for L2VNI localnet ports owned by this agent.

Returns True only if: 1. Port type is localnet 2. Port name follows L2VNI naming: neutron-<uuid>-localnet-<physnet> 3. This agent owns the network (hash ring check)

Note: Event type filtering (CREATE/DELETE only) is handled by the parent BaseEvent.matches() method, which ensures UPDATE events are filtered out before this method is called.

Parameters

- **event** Event type (ROW_CREATE or ROW_DELETE)
- **row** OVN Logical_Switch_Port row
- **old** Previous row state (unused for CREATE events)

Returns

True if event should be processed, False otherwise

run(*event*, *row*, *old*)

Trigger targeted L2VNI reconciliation for specific VLAN.

Parameters

- **event** Event type (ROW_CREATE or ROW_DELETE)
- **row** OVN Logical_Switch_Port row
- **old** Previous row state (used for DELETE events)

table: str = 'Logical_Switch_Port'

networking_baremetal.agent.router_ha_binding module

Router HA Binding Manager.

Manages HA chassis group binding for router interface ports on VLAN networks with baremetal nodes. Ensures router interface ports are bound to the same HA chassis group as the networks external ports, enabling router-to-baremetal communication on physical networks.

This is related to LP#1995078 where baremetal nodes on VLAN networks cannot communicate with their router gateway because the routers internal interface port (LRP) is not bound to any chassis.

This fixes LP#2144458 by providing event-driven HA chassis group binding, eliminating the multi-minute connectivity delays caused by periodic-only reconciliation.

class networking_baremetal.agent.router_ha_binding.**RouterHABindingManager**(*neutron_client*, *ovn_nb_idl*, *member_manager*, *agent_id*)

Bases: object

Manages HA chassis group binding for router interface ports.

Ensures router interface ports on VLAN networks are bound to the same HA chassis group as the networks external ports, enabling router-to- baremetal communication on physical networks.

This manager is event-driven, responding to HA_Chassis_Group creation events to immediately bind router interface ports when network-level HA chassis groups are created.

bind_router_interfaces_for_network(*network_id*, *ha_chassis_group*)

Bind router interface ports to networks HA chassis group.

This is the main entry point called by event handlers when a network HA chassis group is created or updated. It finds all router interface ports on the network and binds them to the specified HA chassis group.

Parameters

- **network_id** Neutron network UUID
- **ha_chassis_group** OVN HA_Chassis_Group UUID or name

reconcile()

Periodic reconciliation of router HA binding.

Discovers all networks with HA chassis groups and ensures their router interface ports are bound to those groups. This catches:

1. Routers added to networks after HA chassis group exists
2. Missed events (agent down/restarting during event)
3. Race conditions or out-of-order event processing
4. Manual changes to LRP ha_chassis_group settings

This method is called periodically (default: 600s / 10 minutes) to ensure eventual consistency even if event-driven binding fails.

Module contents

networking_baremetal.drivers package

Submodules

networking_baremetal.drivers.base module

class networking_baremetal.drivers.base.**BaseDeviceClient**(*device*)

Bases: object

edit_config(*config*)

Edit configuration on the device

Parameters

config The configuration to apply to the device

get(**kwargs*)

Get current configuration/state from device

get_client_args()

Get client connection arguments from configuration

class networking_baremetal.drivers.base.**BaseDeviceDriver**(*device*)

Bases: object

SUPPORTED_BOND_MODES = {}

`create_network`(*context*)

Create network on device

Parameters

context NetworkContext instance describing the new network.

`create_port`(*context*, *segment*, *links*)

Create/Configure port on device

Parameters

- **context** PortContext instance describing the new state of the port, as well as the original state prior to the update_port call.
- **segment** segment dictionary describing segment to bind
- **links** Local link information filtered for the device.

`delete_network`(*context*)

Delete network on device

Parameters

context NetworkContext instance describing the new network.

`delete_port`(*context*, *links*, *current=True*)

Delete/Un-configure port on device

Parameters

- **context** PortContext instance describing the new state of the port, as well as the original state prior to the update_port call.
- **links** Local link information filtered for the device.
- **current** Boolean, when true use context.current, when false use context.original

`load_config`()

Register driver specific configuration

All drivers should register driver specific options in the device specific config group. This method will be called during mechanism driver initialization.

`update_network`(*context*)

Update network on device

Parameters

context NetworkContext instance describing the new network.

`update_port`(*context*, *links*)

Update port on device

Parameters

- **context** PortContext instance describing the new state of the port, as well as the original state prior to the update_port call.
- **links** Local link information filtered for the device.

validate()

Driver validation

This method will be called during mechanism driver initialization. Raising any exception other than DriverValidationError will cause service initialization failure.

Raises

DriverValidationError On validation failure.

Module contents

networking_baremetal.openconfig package

Subpackages

networking_baremetal.openconfig.interfaces package

Submodules

networking_baremetal.openconfig.interfaces.aggregate module

class networking_baremetal.openconfig.interfaces.aggregate.**InterfacesAggregation**

Bases: object

Options for logical interfaces representing aggregates

NAMESPACE = 'http://openconfig.net/yang/interfaces/aggregate'

PARENT = 'interface'

TAG = 'aggregation'

property config

property switched_vlan

to_xml_element()

Create XML Element

Returns

ElementTree Element with SubElements

class networking_baremetal.openconfig.interfaces.aggregate.**InterfacesAggregationConfig**(*opera*
str
=
Net-
con-
fEd-
it-
Con-
fig-
Op-
er-
a-
tion.M

Bases: object

NAMESPACE = 'http://openconfig.net/yang/interfaces/aggregate'

PARENT = 'aggregation'

TAG = 'config'

property lag_type

property min_links

property operation

RFC 6241 - <edit-config> operation attribute

to_xml_element()

Create XML Element

Returns

ElementTree Element with SubElements

networking_baremetal.openconfig.interfaces.ethernet module

class networking_baremetal.openconfig.interfaces.ethernet.InterfacesEthernet

Bases: object

Ethernet configuration and state

NAMESPACE = 'http://openconfig.net/yang/interfaces/ethernet'

PARENT = 'interface'

TAG = 'ethernet'

property config

Configuration parameters for interface

property switched_vlan

to_xml_element()

Create XML Element

Returns

ElementTree Element with SubElements

class networking_baremetal.openconfig.interfaces.ethernet.InterfacesEthernetConfig(*operation=*

Bases: object

OpenConfig interface ethernet configuration

NAMESPACE = 'http://openconfig.net/yang/interfaces'

PARENT = 'interface'

TAG = 'config'

property aggregate_id

Logical aggregate interface for interface

property operation

RFC 6241 - <edit-config> operation attribute

to_xml_element()

Create XML Element

Returns

ElementTree Element with SubElements

networking_baremetal.openconfig.interfaces.interfaces module

class networking_baremetal.openconfig.interfaces.interfaces.**BaseInterface**(*name: str*)

Bases: object

Base interface

NAMESPACE = 'http://openconfig.net/yang/interfaces'

PARENT = 'interfaces'

TAG = 'interface'

property config

Configuration parameters for interface

property name

The name of the interface.

to_xml_element()

Create XML Element

Returns

ElementTree Element with SubElements

class networking_baremetal.openconfig.interfaces.interfaces.**InterfaceAggregate**(*name: str*, *operation: str = NetconfEditConfigOperation.MERGE*)

Bases: *BaseInterface*

property aggregation

Ethernet configuration and state

property operation

RFC 6241 - <edit-config> operation attribute

to_xml_element()

Create XML Element

Returns

ElementTree Element with SubElements

class networking_baremetal.openconfig.interfaces.interfaces.**InterfaceConfig**(*operation=NetconfE*
name: str | None = None, description: str | None = None, enabled: bool | None = None, mtu: int | None = None)

Bases: object

OpenConfig interface configuration

NAMESPACE = 'http://openconfig.net/yang/interfaces'

PARENT = 'interface'

TAG = 'config'

property description

A textual description of the interface

property enabled

The configured, desired state of the interface

property mtu

The max transmission unit size in octets

property name

The name of the interface.

property operation

RFC 6241 - <edit-config> operation attribute

to_xml_element()

Create XML Element

Returns

ElementTree Element with SubElements

class networking_baremetal.openconfig.interfaces.interfaces.**InterfaceEthernet**(*name: str*)

Bases: *BaseInterface*

property ethernet

Ethernet configuration and state

to_xml_element()

Create XML Element

Returns

ElementTree Element with SubElements

class networking_baremetal.openconfig.interfaces.interfaces.**Interfaces**

Bases: Collection

Group/List of interfaces

NAMESPACE = 'http://openconfig.net/yang/interfaces'

TAG = 'interfaces'

add(*name: str*, *interface_type: str = 'ethernet'*)

Add interface

Parameters

- **name** Interface name
- **interface_type** Interface type (ethernet, aggregate, base)

Type

str

Type

str

property interfaces

List of interfaces

to_xml_element()

Create XML Element

Returns

ElementTree Element with SubElements

networking_baremetal.openconfig.interfaces.types module

class networking_baremetal.openconfig.interfaces.types.**AggregationType**(*value*, *names=<not given>*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

LACP = 'LACP'

STATIC = 'SATIC'

Module contents

networking_baremetal.openconfig.lacp package

Submodules

networking_baremetal.openconfig.lacp.lacp module

class networking_baremetal.openconfig.lacp.lacp.**LACP**

Bases: object

LACP Top level

LACP configuration and state variable containers

NAMESPACE = 'http://openconfig.net/yang/lacp'

TAG = 'lacp'

property interfaces

to_xml_element()

Create XML Element

Returns

ElementTree Element with SubElements

class networking_baremetal.openconfig.lacp.lacp.**LACPInterface**(*name: str*, *operation=NetconfEditConfigOperation.ME*

Bases: object

Base LACP aggregate interface

NAMESPACE = 'http://openconfig.net/yang/lacp'

PARENT = 'interfaces'

TAG = 'interface'

property config

Configuration data for each LACP aggregate interface

property name

The name of the LACP aggregate interface.

property operation

RFC 6241 - <edit-config> operation attribute

to_xml_element()

Create XML Element

Returns

ElementTree Element with SubElements

class networking_baremetal.openconfig.lacp.lacp.**LACPInterfaceConfig**(*name: str*, *operation=NetconfEditConfigOpera*
interval=LACPPeriod.SLOW, *lacp_mode=LACPActivity.ACT*

Bases: object

OpenConfig LACP aggregate interface configuration

NAMESPACE = 'http://openconfig.net/yang/lacp'

PARENT = 'interface'

TAG = 'config'

property interval

The period between LACP messages

property lacp_mode

The LACP mode if the aggregate interface

property name

The name of the interface.

property operation

RFC 6241 - <edit-config> operation attribute

to_xml_element()

Create XML Element

Returns
ElementTree Element with SubElements

class networking_baremetal.openconfig.lacp.lacp.**LACPInterfaces**

Bases: Collection

Top-level grouping for LACP-enabled interfaces

NAMESPACE = 'http://openconfig.net/yang/lacp'

PARENT = 'lacp'

TAG = 'interfaces'

add(*name: str*)

Add interface

Parameters
name Interface name

Type
str

property interfaces

List of interfaces

to_xml_element()

Create XML Element

Returns
ElementTree Element with SubElements

networking_baremetal.openconfig.lacp.types module

class networking_baremetal.openconfig.lacp.types.**LACPActivity**(*value*, *names=<not given>*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

Describes the LACP membership type

Active or passive, of the interface in the aggregate. reference IEEE 802.1AX-2008

ACTIVE: Interface is an active member, i.e., will detect and
maintain aggregates

PASSIVE: Interface is a passive member, i.e., it participates
with an active partner

ACTIVE = 'ACTIVE'

PASSIVE = 'PASSIVE'

class networking_baremetal.openconfig.lacp.types.**LACPPeriod**(*value*, *names=<not given>*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

Defines the time between sending LACP messages

reference IEEE 802.3ad FAST: Send LACP packets every second SLOW: Send LACP packets every 30 seconds

FAST = 'FAST'

SLOW = 'SLOW'

Module contents

networking_baremetal.openconfig.network_instance package

Submodules

networking_baremetal.openconfig.network_instance.network_instance module

class networking_baremetal.openconfig.network_instance.network_instance.**NetworkInstance**(*nam*

Bases: object

An OpenConfig description of a network_instance.

This may be a Layer 3 forwarding construct such as a virtual routing and forwarding (VRF) instance, or a Layer 2 instance such as a virtual switch instance (VSI). Mixed Layer 2 and Layer 3 instances are also supported.

NAMESPACE = 'http://openconfig.net/yang/network-instance'

TAG = 'network-instance'

property name

A unique name identifying the network instance

to_xml_element()

Create XML Element

Returns

ElementTree Element with SubElements

property vlans

Group/List of VLANs - keyed by id

class networking_baremetal.openconfig.network_instance.network_instance.**NetworkInstances**

Bases: Collection

Top-level grouping containing a list of network instances.

NAMESPACE = 'http://openconfig.net/yang/network-instance'

TAG = 'network-instances'

add(*name: str*)

Add network instance

Parameters

name A unique name identifying the network instance

Type

str

Keyword arguments

Network instance arguments

property network_instances

to_xml_element()

Create XML Element

Returns

ElementTree Element with SubElements

Module contents

networking_baremetal.openconfig.vlan package

Submodules

networking_baremetal.openconfig.vlan.types module

class networking_baremetal.openconfig.vlan.types.**VlanId**(*vlan_id: int*)

Bases: object

Type definition representing a single-tagged VLAN

property vlan_id

class networking_baremetal.openconfig.vlan.types.**VlanInterfaceMode**(*value*, *names=<not given>*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

VLAN interface mode (trunk or access)

ACCESS = 'ACCESS'

TRUNK = 'TRUNK'

class networking_baremetal.openconfig.vlan.types.**VlanRange**(*vlan_range: str*)

Bases: object

Type definition representing a range of single-tagged VLANs.

A range is specified as x..y where x and y are valid VLAN IDs (1 <= vlan-id <= 4094). The range is assumed to be inclusive, such that any VLAN-ID matching x <= VLAN-ID <= y falls within the range.

pattern =
re.compile('^(409[0-4]|40[0-8][0-9]|[1-3][0-9]{3}|[1-9][0-9]{1,2}|[1-9])
.\\.(409[0-4]|40[0-8][0-9]|[1-3][0-9]{3}|[1-9][0-9]{1,2}|[1-9])$')

property vlan_range

class networking_baremetal.openconfig.vlan.types.**VlanStatus**(*value*, *names=<not given>*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

VLAN Admin state

ACTIVE: VLAN is active SUSPENDED: VLAN is inactive / suspended

ACTIVE = 'ACTIVE'

SUSPENDED = 'SUSPENDED'

networking_baremetal.openconfig.vlan.vlan module

class networking_baremetal.openconfig.vlan.vlan.**TrunkVlans**

Bases: Collection

add(*value*)

Add vlan or range of vlans (range: 100..200)

class networking_baremetal.openconfig.vlan.vlan.**Vlan**(*vlan_id: int*, *operation=NetconfEditConfigOperation.MERGE*)

Bases: object

Base vlan

NAMESPACE = 'http://openconfig.net/yang/vlan'

PARENT = 'vlans'

TAG = 'vlan'

property config

Configuration parameters for VLAN

property operation

RFC 6241 - <edit-config> operation attribute

to_xml_element()

Create XML Element

Returns

ElementTree Element with SubElements

property vlan_id

The id of the VLAN

class networking_baremetal.openconfig.vlan.vlan.**VlanConfig**(*operation=NetconfEditConfigOperation.M*
vlan_id: int = None,
name: str = None, *status:*
str = None)

Bases: object

OpenConfig VLAN configuration

NAMESPACE = 'http://openconfig.net/yang/vlan'

PARENT = 'vlan'

TAG = 'config'

property name

Interface VLAN name.

property operation

RFC 6241 - <edit-config> operation attribute

property status

Admin state of the VLAN

to_xml_element()

Create XML Element

Returns

ElementTree Element with SubElements

property vlan_id

The id of the VLAN

class networking_baremetal.openconfig.vlan.vlan.**VlanSwitchedConfig**(*operation: str*
= NetconfEdit-
ConfigOpera-
tion.MERGE,
inter-
face_mode: str
| None = None,
native_vlan:
int | None =
None,
access_vlan:
int | None =
None)

Bases: object

Ethernet interface VLAN config

VLAN related configuration that is part of the physical Ethernet interface.

NAMESPACE = 'http://openconfig.net/yang/vlan'

PARENT = 'switched-vlan'

TAG = 'config'

property access_vlan

Access VLAN assigned to the interfaces

property interface_mode

Get the interface to access or trunk mode for VLANs

property native_vlan

Native VLAN

is valid for trunk mode interfaces

property operation

RFC 6241 - <edit-config> operation attribute

to_xml_element()

Create XML Element

Returns

ElementTree Element with SubElements

property trunk_vlans

Allowed VLANs may be specified for trunk mode interfaces

class networking_baremetal.openconfig.vlan.vlan.**VlanSwitchedVlan**

Bases: object

VLAN interface-specific data on Ethernet interfaces.

Enclosing container for VLAN interface-specific data on Ethernet interfaces. These are for standard L2, switched-style VLANs.

NAMESPACE = 'http://openconfig.net/yang/vlan'

PARENT = 'ethernet'

TAG = 'switched-vlan'

property config

Configuration parameters for VLANs

to_xml_element()

Create XML Element

Returns

ElementTree Element with SubElements

class networking_baremetal.openconfig.vlan.vlan.**Vlans**

Bases: Collection

Group/List of VLANs

NAMESPACE = 'http://openconfig.net/yang/vlan'

TAG = 'vlans'

add(*vlan_id: int*)

Add VLAN

Parameters
vlan_id VLAN ID

Type
int

Keyword arguments
VLAN configuration

remove(*vlan_id: int*)

Remove VLAN

Parameters
vlan_id VLAN ID

Type
int

to_xml_element()

Create XML Element

Returns
ElementTree Element with SubElements

property vlans

List of VLANs

Module contents

Module contents

networking_baremetal.plugins package

Subpackages

networking_baremetal.plugins.ml2 package

Submodules

networking_baremetal.plugins.ml2.baremetal_l2vni_mapping module

class networking_baremetal.plugins.ml2.baremetal_l2vni_mapping.**L2vniMechanismDriver**

Bases: MechanismDriver

ML2 mechanism driver for L2VNI binding

This mechanism driver is called on port binding to facilitate the VTEP to VLAN binding necessary for EVPN networks to attach to baremetal ports, which may then connect to the environment through an EVPN connection, or through direct port attachments

bind_port(*context*)

Attempt to bind a port.

Parameters
context PortContext instance describing the port

This method is called outside any transaction to attempt to establish a port binding using this mechanism driver. Bindings may be created at each of multiple levels of a hierarchical network, and are established from the top level downward. At each level, the mechanism driver determines whether it can bind to any of the network segments in the context.segments_to_bind property, based on the value of the context.host property, any relevant port or network attributes, and its own knowledge of the network topology. At the top level, context.segments_to_bind contains the static segments of the ports network. At each lower level of binding, it contains static or dynamic segments supplied by the driver that bound at the level above. If the driver is able to complete the binding of the port to any segment in context.segments_to_bind, it must call context.set_binding with the binding details. If it can partially bind the port, it must call context.continue_binding with the network segments to be used to bind at the next lower level.

If the binding results are committed after bind_port returns, they will be seen by all mechanism drivers as update_port_precommit and update_port_postcommit calls. But if some other thread or process concurrently binds or updates the port, these binding results will not be committed, and update_port_precommit and update_port_postcommit will not be called on the mechanism drivers with these results. Because binding results can be discarded rather than committed, drivers should avoid making persistent state changes in bind_port, or else must ensure that such state changes are eventually cleaned up.

Implementing this method explicitly declares the mechanism driver as having the intention to bind ports. This is inspected by the QoS service to identify the available QoS rules you can use with ports.

property connectivity

Return the mechanism driver connectivity type

The possible values are l2, l3 and legacy (default).

Returns
a string in (l2, l3, legacy)

delete_port_postcommit(*context*)

Clean up localnet port when last baremetal port is deleted.

When a baremetal port is deleted, check if it was the last port using the dynamic VLAN segment. If so, remove the localnet port and release the segment to prevent VLAN ID reuse conflicts.

get_allowed_network_types(*agent*)

Return the agents or drivers allowed network types.

L2VNI handles hierarchical port binding for overlay networks only. Returns vxlan and geneve, which are handled by creating dynamic VLAN segments. Flat and VLAN networks are handled by the base baremetal mechanism driver.

initialize()

Perform driver initialization.

Called after all drivers have been loaded and the database has been initialized. No abstract methods defined below will be called prior to this method being called.

update_port_postcommit(*context*)

Update a port.

Parameters
context PortContext instance describing the new state of the port, as well as the original state prior to the update_port call.

Called after the transaction completes. Call can block, though will block the entire process so care should be taken to not drastically affect performance. Raising an exception will result in the deletion of the resource.

update_port_postcommit is called for all changes to the port state. It is up to the mechanism driver to ignore state or state changes that it does not know or care about.

networking_baremetal.plugins.ml2.baremetal_mech module

class networking_baremetal.plugins.ml2.baremetal_mech.**BaremetalMechanismDriver**

Bases: SimpleAgentMechanismDriverBase

bind_port(*context*)

Bind port, skipping hierarchical overlayVLAN scenarios.

Check if this is a hierarchical binding scenario where an overlay segment exists on the network. If so, skip binding to allow other drivers like genericswitch to handle the VLAN binding.

property connectivity

Return the mechanism driver connectivity type

The possible values are l2, l3 and legacy (default).

Returns
a string in (l2, l3, legacy)

create_network_postcommit(*context*)

Create a network.

Called after the transaction commits. Call can block, though will block the entire process so care should be taken to not drastically affect performance. Raising an exception will cause the deletion of the resource.

Parameters
context NetworkContext instance describing the new network.

create_network_precommit(*context*)

Allocate resources for a new network.

Create a new network, allocating resources as necessary in the database. Called inside transaction context on session. Call cannot block. Raising an exception will result in a rollback of the current transaction.

Parameters
context NetworkContext instance describing the new network.

create_port_postcommit(*context*)

Create a port.

Called after the transaction completes. Call can block, though will block the entire process so care should be taken to not drastically affect performance. Raising an exception will result in the deletion of the resource.

Parameters
context PortContext instance describing the port.

create_port_precommit(*context*)

Allocate resources for a new port.

Create a new port, allocating resources as necessary in the database. Called inside transaction context on session. Call cannot block. Raising an exception will result in a rollback of the current transaction.

Parameters
context PortContext instance describing the port.

create_subnet_postcommit(*context*)

Create a subnet.

Called after the transaction commits. Call can block, though will block the entire process so care should be taken to not drastically affect performance. Raising an exception will cause the deletion of the resource.

Parameters
context SubnetContext instance describing the new subnet.

create_subnet_precommit(*context*)

Allocate resources for a new subnet.

Create a new subnet, allocating resources as necessary in the database. Called inside transaction context on session. Call cannot block. Raising an exception will result in a rollback of the current transaction.

Parameters
context SubnetContext instance describing the new subnet.

delete_network_postcommit(*context*)

Delete a network.

Called after the transaction commits. Call can block, though will block the entire process so care should be taken to not drastically affect performance. Runtime errors are not expected, and will not prevent the resource from being deleted.

Parameters
context NetworkContext instance describing the current state of the network, prior to the call to delete it.

delete_network_precommit(*context*)

Delete resources for a network.

Delete network resources previously allocated by this mechanism driver for a network. Called inside transaction context on session. Runtime errors are not expected, but raising an exception will result in rollback of the transaction.

Parameters
context NetworkContext instance describing the current state of the network, prior to the call to delete it.

delete_port_postcommit(*context*)

Delete a port.

state of the port, prior to the call to delete it. Called after the transaction completes. Call can block, though will block the entire process so care should be taken to not drastically affect performance. Runtime errors are not expected, and will not prevent the resource from being deleted.

Parameters
context PortContext instance describing the current state of the port, prior to the call to delete it.

delete_port_precommit(*context*)

Delete resources of a port.

Called inside transaction context on session. Runtime errors are not expected, but raising an exception will result in rollback of the transaction.

Parameters
context PortContext instance describing the current state of the port, prior to the call to delete it.

delete_subnet_postcommit(*context*)

Delete a subnet.a

Called after the transaction commits. Call can block, though will block the entire process so care should be taken to not drastically affect performance. Runtime errors are not expected, and will not prevent the resource from being deleted.

Parameters
context SubnetContext instance describing the current state of the subnet, prior to the call to delete it.

delete_subnet_precommit(*context*)

Delete resources for a subnet.

Delete subnet resources previously allocated by this mechanism driver for a subnet. Called inside transaction context on session. Runtime errors are not expected, but raising an exception will result in rollback of the transaction.

Parameters
context SubnetContext instance describing the current state of the subnet, prior to the call to delete it.

get_allowed_network_types(*agent*)

Return the agents or drivers allowed network types.

For example: return (flat,). You can also refer to the configuration the given agent exposes.

`get_mappings`(*agent*)

Return the agents bridge or interface mappings.

For example: agent[configurations].get(bridge_mappings, {}).

`try_to_bind_segment_for_agent`(*context*, *segment*, *agent*)

Try to bind with segment for agent.

Parameters

- **context** PortContext instance describing the port
- **segment** segment dictionary describing segment to bind
- **agent** agents_db entry describing agent to bind

Returns

True iff segment has been bound for agent

Neutron segments api-ref:
https://docs.openstack.org/api-ref/network/v2/#segments

Example segment dictionary: {segmentation_id: segmentation_id,
network_type: network_type, id: segment_uuid}

Called outside any transaction during bind_port() so that derived MechanismDrivers can use agent_db data along with built-in knowledge of the corresponding agents capabilities to attempt to bind to the specified network segment for the agent.

If the segment can be bound for the agent, this function must call context.set_binding() with appropriate values and then return True. Otherwise, it must return False.

`update_network_postcommit`(*context*)

Update a network.

Called after the transaction commits. Call can block, though will block the entire process so care should be taken to not drastically affect performance. Raising an exception will cause the deletion of the resource. update_network_postcommit is called for all changes to the network state. It is up to the mechanism driver to ignore state or state changes that it does not know or care about.

Parameters

context NetworkContext instance describing the new state of the network, as well as the original state prior to the update_network call.

`update_network_precommit`(*context*)

Update resources of a network.

Update values of a network, updating the associated resources in the database. Called inside transaction context on session. Raising an exception will result in rollback of the transaction. update_network_precommit is called for all changes to the network state. It is up to the mechanism driver to ignore state or state changes that it does not know or care about.

Parameters

context NetworkContext instance describing the new state of the network, as well as the original state prior to the update_network call.

update_port_postcommit(*context*)

Update a port.

Called after the transaction completes. Call can block, though will block the entire process so care should be taken to not drastically affect performance. Raising an exception will result in the deletion of the resource. update_port_postcommit is called for all changes to the port state. It is up to the mechanism driver to ignore state or state changes that it does not know or care about.

Parameters

context PortContext instance describing the new state of the port, as well as the original state prior to the update_port call.

update_port_precommit(*context*)

Update resources of a port.

Called inside transaction context on session to complete a port update as defined by this mechanism driver. Raising an exception will result in rollback of the transaction. update_port_precommit is called for all changes to the port state. It is up to the mechanism driver to ignore state or state changes that it does not know or care about.

Parameters

context PortContext instance describing the new state of the port, as well as the original state prior to the update_port call.

update_subnet_postcommit(*context*)

Update a subnet.

Called after the transaction commits. Call can block, though will block the entire process so care should be taken to not drastically affect performance. Raising an exception will cause the deletion of the resource. update_subnet_postcommit is called for all changes to the subnet state. It is up to the mechanism driver to ignore state or state changes that it does not know or care about.

Parameters

context SubnetContext instance describing the new state of the subnet, as well as the original state prior to the update_subnet call.

update_subnet_precommit(*context*)

Update resources of a subnet.

Update values of a subnet, updating the associated resources in the database. Called inside transaction context on session. Raising an exception will result in rollback of the transaction. update_subnet_precommit is called for all changes to the subnet state. It is up to the mechanism driver to ignore state or state changes that it does not know or care about.

Parameters

context SubnetContext instance describing the new state of the subnet, as well as the original state prior to the update_subnet call.

Module contents

Module contents

Submodules

networking_baremetal.common module

networking_baremetal.common.**config_to_xml**(*config*)

networking_baremetal.common.**driver_mgr**(*device_id*)

networking_baremetal.common.**txt_subelement**(*parent*, *tag*, *text*, **args*, ***kwargs*)

networking_baremetal.config module

networking_baremetal.config.**get_devices**()

Get enabled network devices from configuration

This is called during driver initialization, during initialization additional driver specific configuration is loaded and the drivers validation method is called.

networking_baremetal.config.**list_common_device_driver_opts**()

networking_baremetal.config.**list_opts**()

networking_baremetal.constants module

class networking_baremetal.constants.**NetconfEditConfigOperation**(*value*, *names=<not given>*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

RFC 6241 - <edit-config> operation attribute

The operation attribute has one of the following values:

merge: The configuration data identified by the element
containing this attribute is merged with the configuration at the corresponding level in the configuration datastore identified by the <target> parameter. This is the default behavior.

replace: The configuration data identified by the element
containing this attribute replaces any related configuration in the configuration datastore identified by the <target> parameter. If no such configuration data exists in the configuration datastore, it is created. Unlike a <copy-config> operation, which replaces the entire target configuration, only the configuration actually present in the <config> parameter is affected.

create: The configuration data identified by the element
containing this attribute is added to the configuration if and only if the configuration data does not already exist in the configuration datastore. If the configuration data exists, an <rpc-error> element is returned with an <error-tag> value of data-exists.

delete: The configuration data identified by the element
containing this attribute is deleted from the configuration if and only if the configuration data currently exists in the configuration datastore. If the configuration data does not exist, an <rpc-error> element is returned with an <error-tag> value of data-missing.

remove: The configuration data identified by the element
containing this attribute is deleted from the configuration if the configuration data currently exists in the configuration datastore. If the configuration data does not exist, the remove operation is silently ignored by the server.

CREATE = 'create'

DELETE = 'delete'

MERGE = 'merge'

REMOVE = 'remove'

REPLACE = 'replace'

networking_baremetal.exceptions module

exception networking_baremetal.exceptions.**DeviceConnectionError**(***kwargs*)

Bases: NeutronException

message = 'Driver failed connecting to device %(device)s: %(err)s'

exception networking_baremetal.exceptions.**DriverEntrypointLoadError**(***kwargs*)

Bases: NeutronException

message = 'Failed to load entrypoint %(entry_point)s: %(err)s'

exception networking_baremetal.exceptions.**DriverValidationError**(***kwargs*)

Bases: NeutronException

message = 'Failed driver validation for device %(device)s: %(err)s'

exception networking_baremetal.exceptions.**PreConfiguredAggrergateNotFound**(***kwargs*)

Bases: NeutronException

message = 'Driver could not find the aggregate ID for the pre-configured link aggregate for links %(links)s on device %(device)s.'

networking_baremetal.ironic_client module

networking_baremetal.ironic_client.**get_client**()

Get an ironic client connection.

networking_baremetal.ironic_client.**get_session**(*group*)

networking_baremetal.ironic_client.**list_opts**()

networking_baremetal.neutron_client module

networking_baremetal.neutron_client.**get_client**()

Get a Neutron client connection via OpenStack SDK.

Returns
OpenStack SDK Connection object for accessing network APIs

networking_baremetal.neutron_client.**get_session**(*group*)

Get a session for the specified config group.

Parameters

group Configuration group name

Returns

keystoneauth1 session

networking_baremetal.neutron_client.**list_opts**()

Return neutron client configuration options.

Module contents

CHAPTER

SEVEN

RELEASE NOTES

Release Notes

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

INDEX

D

E

F

G

H

I

L

M

N

O

U

V

W