NAME

ovn-nb − OVN_Northbound database schema

This database is the interface between OVN and the cloud management system (CMS), such as OpenStack, running above it. The CMS produces almost all of the contents of the database. The **ovn−northd** program monitors the database contents, transforms it, and stores it into the **OVN_Southbound** database.

We generally speak of ‘‘the’’ CMS, but one can imagine scenarios in which multiple CMSes manage different parts of an OVN deployment.

External IDs

Each of the tables in this database contains a special column, named **external_ids**. This column has the same form and purpose each place it appears.

external_ids: map of string-string pairs
Key-value pairs for use by the CMS. The CMS might use certain pairs, for example, to identify entities in its own configuration that correspond to those in this database.

TABLE SUMMARY

The following list summarizes the purpose of each of the tables in the **OVN_Northbound** database. Each table is described in more detail on a later page.

Table	Purpose
NB_Global	Northbound configuration
Logical_Switch	L2 logical switch
Logical_Switch_Port	L2 logical switch port
Forwarding_Group	forwarding group
Address_Set	Address Sets
Port_Group	Port Groups
Load_Balancer	load balancer
Load_Balancer_Health_Check	load balancer
ACL	Access Control List (ACL) rule
Logical_Router	L3 logical router
QoS	QoS rule
Meter	Meter entry
Meter_Band	Band for meter entries
Logical_Router_Port	L3 logical router port
Logical_Router_Static_Route	Logical router static routes
Logical_Router_Policy	Logical router policies
NAT	NAT rules
DHCP_Options	DHCP options
Connection	OVSDB client connections.
DNS	Native DNS resolution
SSL	SSL configuration.
Gateway_Chassis	Gateway_Chassis configuration.
HA_Chassis_Group	HA_Chassis_Group configuration.
HA_Chassis	HA_Chassis configuration.
BFD	BFD configuration.

NB_Global TABLE

Northbound configuration for an OVN system. This table must have exactly one row.

Summary:

Identity:	
name	string
Status:	
nb_cfg	integer
nb_cfg_timestamp	integer
sb_cfg	integer
sb_cfg_timestamp	integer
hv_cfg	integer
hv_cfg_timestamp	integer
Common Columns:	
external_ids	map of string-string pairs
Common options:	
options	map of string-string pairs
Options for configuring OVS BFD:	
options : bfd-min-rx	optional string
options : bfd-decay-min-rx	optional string
options : bfd-min-tx	optional string
options : bfd-mult	optional string
options : mac_prefix	optional string
options : controller_event	optional string, either **true** or **false**
options : northd_probe_interval	optional string
options : use_logical_dp_groups	optional string
options : ignore_lsp_down	optional string
Options for configuring interconnection route advertisement:	
options : ic-route-adv	optional string
options : ic-route-learn	optional string
options : ic-route-adv-default	optional string
options : ic-route-learn-default	optional string
options : ic-route-blacklist	optional string
Connection Options:	
connections	set of **Connection**s
ssl	optional **SSL**
Security Configurations:	
ipsec	boolean
Read-only Options:	
options : max_tunid	optional string

Details:

Identity:

name: string

The name of the OVN cluster, which uniquely identifies the OVN cluster throughout all OVN clusters supposed to interconnect with each other.

Status:

These columns allow a client to track the overall configuration state of the system.

nb_cfg: integer

Sequence number for client to increment. When a client modifies any part of the northbound database configuration and wishes to wait for **ovn−northd** and possibly all of the hypervisors to finish applying the changes, it may increment this sequence number.

nb_cfg_timestamp: integer
The timestamp, in milliseconds since the epoch, when **ovn−northd** sees the latest **nb_cfg** and starts processing.

To print the timestamp as a human-readable date:

date −d "@$(ovn−nbctl get NB_Global . nb_cfg_timestamp | sed 's/...$//')"

sb_cfg: integer
Sequence number that **ovn−northd** sets to the value of **nb_cfg** after it finishes applying the corresponding configuration changes to the **OVN_Southbound** database.

sb_cfg_timestamp: integer
The timestamp, in milliseconds since the epoch, when **ovn−northd** finishes applying the corresponding configuration changes to the **OVN_Southbound** database successfully.

hv_cfg: integer
Sequence number that **ovn−northd** sets to the smallest sequence number of all the chassis in the system, as reported in the **Chassis_Private** table in the southbound database. Thus, **hv_cfg** equals **nb_cfg** if all chassis are caught up with the northbound configuration (which may never happen, if any chassis is down). This value can regress, if a chassis was removed from the system and rejoins before catching up.

If there are no chassis, then **ovn−northd** copies **nb_cfg** to **hv_cfg**. Thus, in this case, the (nonexistent) hypervisors are always considered to be caught up. This means that hypervisors can be "caught up" even in cases where **sb_cfg** would show that the southbound database is not. To detect when both the hypervisors and the southbound database are caught up, a client should take the smaller of **sb_cfg** and **hv_cfg**.

hv_cfg_timestamp: integer
The largest timestamp, in milliseconds since the epoch, of the smallest sequence number of all the chassis in the system, as reported in the **Chassis_Private** table in the southbound database. In other words, this timestamp reflects the time when the slowest chassis catches up with the northbound configuration, which is useful for end-to-end control plane latency measurement.

Common Columns:

external_ids: map of string-string pairs
See **External IDs** at the beginning of this document.

Common options:

options: map of string-string pairs
This column provides general key/value settings. The supported options are described individually below.

Options for configuring OVS BFD:

These options apply when **ovn−controller** configures OVS BFD on tunnels interfaces. Please note these parameters refer to legacy OVS BFD implementation and not to OVN BFD one.

options : bfd-min-rx: optional string
BFD option **min−rx** value to use when configuring BFD on tunnel interfaces.

options : bfd-decay-min-rx: optional string
BFD option **decay−min−rx** value to use when configuring BFD on tunnel interfaces.

options : bfd-min-tx: optional string
BFD option **min−tx** value to use when configuring BFD on tunnel interfaces.

options : bfd-mult: optional string
BFD option **mult** value to use when configuring BFD on tunnel interfaces.

options : mac_prefix: optional string
Configure a given OUI to be used as prefix when L2 address is dynamically assigned, e.g. **00:11:22**

options : controller_event: optional string, either **true** or **false**
Value set by the CMS to enable/disable ovn-controller event reporting. Traffic into OVS can raise a 'controller' event that results in a Controller_Event being written to the **Controller_Event** table in SBDB. When the CMS has seen the event and taken appropriate action, it can remove the corresponding row in **Controller_Event** table. The intention is for a CMS to see the events and take some sort of action. Please see the **Controller_Event** table in SBDB. It is possible to associate a meter to each controller event type in order to not overload the pinctrl thread under heavy load. Each event type relies on a meter with a defined name:

- empty_lb_backends: event-elb

options : northd_probe_interval: optional string
The inactivity probe interval of the connection to the OVN Northbound and Southbound databases from **ovn−northd**, in milliseconds. If the value is zero, it disables the connection keepalive feature.

If the value is nonzero, then it will be forced to a value of at least 1000 ms.

options : use_logical_dp_groups: optional string
If set to **true**, **ovn−northd** will combine logical flows that differs only by logical datapath into a single logical flow with logical datapath group attached.

While this should significantly reduce number of logical flows stored in Southbound database this could also increase processing complexity on the **ovn−controller** side, e.g., **ovn−controller** will re-consider logical flow for all logical datapaths in a group. If the option set to **false**, there will be separate logical flow per logical datapath and only this flow will be re-considered.

The default value is **false**.

options : ignore_lsp_down: optional string
If set to false, ARP/ND reply flows for logical switch ports will be installed only if the port is up, i.e. claimed by a Chassis. If set to true, these flows are installed regardless of the status of the port, which can result in a situation that ARP request to an IP is resolved even before the relevant VM/container is running. For environments where this is not an issue, setting it to **true** can reduce the load and latency of the control plane. The default value is **false**.

Options for configuring interconnection route advertisement:

These options control how routes are advertised between OVN deployments for interconnection. If enabled, **ovn−ic** from different OVN deployments exchanges routes between each other through the global **OVN_IC_Southbound** database. Only routers with ports connected to interconnection transit switches participate in route advertisement. For each of these routers, there are two types of routes to be advertised:

Firstly, the static routes configured in the router are advertised.

Secondly, the **networks** configured in the logical router ports that are not on the transit switches are advertised. These are considered as directly connected subnets on the router.

Link local prefixes (IPv4 169.254.0.0/16 and IPv6 FE80::/10) are never advertised.

The learned routes are added to the **static_routes** column of the **Logical_Router** table, with **external_ids:ic−learned−route** set to the uuid of the row in **Route** table of the **OVN_IC_Southbound** database.

options : ic-route-adv: optional string
A boolean value that enables route advertisement to the global **OVN_IC_Southbound** database. Default is **false**.

options : ic-route-learn: optional string
A boolean value that enables route learning from the global **OVN_IC_Southbound** database. Default is **false**.

options : ic-route-adv-default: optional string
A boolean value that enables advertising default route to the global **OVN_IC_Southbound** database. Default is **false**. This option takes effect only when option **ic−route−adv** is **true**.

options : ic-route-learn-default: optional string
A boolean value that enables learning default route from the global **OVN_IC_Southbound** database. Default is **false**. This option takes effect only when option **ic−route−learn** is **true**.

options : ic-route-blacklist: optional string
A string value contains a list of CIDRs delimited by ",". A route will not be advertised or learned if the route's prefix belongs to any of the CIDRs listed.

Connection Options:

connections: set of **Connection**s
Database clients to which the Open vSwitch database server should connect or on which it should listen, along with options for how these connections should be configured. See the **Connection** table for more information.

ssl: optional **SSL**
Global SSL configuration.

Security Configurations:

ipsec: boolean
Tunnel encryption configuration. If this column is set to be true, all OVN tunnels will be encrypted with IPsec.

Read-only Options:

options : max_tunid: optional string
The maximum supported tunnel ID. Depends on types of encapsulation enabled in the cluster.

Logical_Switch TABLE

Each row represents one L2 logical switch.

There are two kinds of logical switches, that is, ones that fully virtualize the network (overlay logical switches) and ones that provide simple connectivity to physical networks (bridged logical switches). They work in the same way when providing connectivity between logical ports on same chassis, but differently when connecting remote logical ports. Overlay logical switches connect remote logical ports by tunnels, while bridged logical switches provide connectivity to remote ports by bridging the packets to directly connected physical L2 segments with the help of **localnet** ports. Each bridged logical switch has one or more **localnet** ports, which have only one special address **unknown**.

Summary:

ports	set of **Logical_Switch_Port**s
load_balancer	set of weak reference to **Load_Balancer**s
acls	set of **ACL**s
qos_rules	set of **QoS**es
dns_records	set of weak reference to **DNS**es
forwarding_groups	set of **Forwarding_Group**s
Naming:	
name	string
external_ids : neutron:network_name	optional string
IP Address Assignment:	
other_config : subnet	optional string
other_config : exclude_ips	optional string
other_config : ipv6_prefix	optional string
other_config : mac_only	optional string, either **true** or **false**
IP Multicast Snooping Options:	
other_config : mcast_snoop	optional string, either **true** or **false**
other_config : mcast_querier	optional string, either **true** or **false**
other_config : mcast_flood_unregistered	optional string, either **true** or **false**
other_config : mcast_table_size	optional string, containing an integer, in range 1 to 32,766
other_config : mcast_idle_timeout	optional string, containing an integer, in range 15 to 3,600
other_config : mcast_query_interval	optional string, containing an integer, in range 1 to 3,600
other_config : mcast_query_max_response	optional string, containing an integer, in range 1 to 10
other_config : mcast_eth_src	optional string
other_config : mcast_ip4_src	optional string
other_config : mcast_ip6_src	optional string
Interconnection:	
other_config : interconn-ts	optional string
Tunnel Key:	
other_config : requested-tnl-key	optional string, containing an integer, in range 1 to 16,777,215
Other options:	
other_config : vlan-passthru	optional string, either **true** or **false**
Common Columns:	
external_ids	map of string-string pairs

Details:

ports: set of **Logical_Switch_Port**s

The logical ports connected to the logical switch.

It is an error for multiple logical switches to include the same logical port.

load_balancer: set of weak reference to **Load_Balancer**s
Load balance a virtual ip address to a set of logical port endpoint ip addresses.

acls: set of **ACL**s
Access control rules that apply to packets within the logical switch.

qos_rules: set of **QoS**es
QoS marking and metering rules that apply to packets within the logical switch.

dns_records: set of weak reference to **DNS**es
This column defines the DNS records to be used for resolving internal DNS queries within the logical switch by the native DNS resolver. Please see the **DNS** table.

forwarding_groups: set of **Forwarding_Group**s
Groups a set of logical port endpoints for traffic going out of the logical switch.

Naming:

These columns provide names for the logical switch. From OVN's perspective, these names have no special meaning or purpose other than to provide convenience for human interaction with the database. There is no requirement for the name to be unique. (For a unique identifier for a logical switch, use its row UUID.)

(Originally, **name** was intended to serve the purpose of a human-friendly name, but the Neutron integration used it to uniquely identify its own switch object, in the format **neutron−***uuid*. Later on, Neutron started propagating the friendly name of a switch as **external_ids:neutron:network_name**. Perhaps this can be cleaned up someday.)

name: string
A name for the logical switch.

external_ids : neutron:network_name: optional string
Another name for the logical switch.

IP Address Assignment:

These options control automatic IP address management (IPAM) for ports attached to the logical switch. To enable IPAM for IPv4, set **other_config:subnet** and optionally **other_config:exclude_ips**. To enable IPAM for IPv6, set **other_config:ipv6_prefix**. IPv4 and IPv6 may be enabled together or separately.

To request dynamic address assignment for a particular port, use the **dynamic** keyword in the **addresses** column of the port's **Logical_Switch_Port** row. This requests both an IPv4 and an IPv6 address, if IPAM for IPv4 and IPv6 are both enabled.

other_config : subnet: optional string
Set this to an IPv4 subnet, e.g. **192.168.0.0/24**, to enable **ovn−northd** to automatically assign IP addresses within that subnet.

other_config : exclude_ips: optional string
To exclude some addresses from automatic IP address management, set this to a list of the IPv4 addresses or **..**-delimited ranges to exclude. The addresses or ranges should be a subset of those in **other_config:subnet**.

Whether listed or not, **ovn−northd** will never allocate the first or last address in a subnet, such as 192.168.0.0 or 192.168.0.255 in 192.168.0.0/24.

Examples:

- **192.168.0.2 192.168.0.10**
- **192.168.0.4 192.168.0.30..192.168.0.60 192.168.0.110..192.168.0.120**
- **192.168.0.110..192.168.0.120 192.168.0.25..192.168.0.30 192.168.0.144**

other_config : ipv6_prefix: optional string
Set this to an IPv6 prefix to enable **ovn−northd** to automatically assign IPv6 addresses using this prefix. The assigned IPv6 address will be generated using the IPv6 prefix and the MAC address

(converted to an IEEE EUI64 identifier) of the port. The IPv6 prefix defined here should be a valid IPv6 address ending with **::**.

Examples:

- **aef0::**
- **bef0:1234:a890:5678::**
- **8230:5678::**

other_config : mac_only: optional string, either **true** or **false**
Value used to request to assign L2 address only if neither subnet nor ipv6_prefix are specified

IP Multicast Snooping Options:

These options control IP Multicast Snooping configuration of the logical switch. To enable IP Multicast Snooping set **other_config:mcast_snoop** to true. To enable IP Multicast Querier set **other_config:mcast_snoop** to true. If IP Multicast Querier is enabled **other_config:mcast_eth_src** and **other_config:mcast_ip4_src** must be set.

other_config : mcast_snoop: optional string, either **true** or **false**
Enables/disables IP Multicast Snooping on the logical switch.

other_config : mcast_querier: optional string, either **true** or **false**
Enables/disables IP Multicast Querier on the logical switch.

other_config : mcast_flood_unregistered: optional string, either **true** or **false**
Determines whether unregistered multicast traffic should be flooded or not. Only applicable if **other_config:mcast_snoop** is enabled.

other_config : mcast_table_size: optional string, containing an integer, in range 1 to 32,766
Number of multicast groups to be stored. Default: 2048.

other_config : mcast_idle_timeout: optional string, containing an integer, in range 15 to 3,600
Configures the IP Multicast Snooping group idle timeout (in seconds). Default: 300 seconds.

other_config : mcast_query_interval: optional string, containing an integer, in range 1 to 3,600
Configures the IP Multicast Querier interval between queries (in seconds). Default: **other_config:mcast_idle_timeout /** 2.

other_config : mcast_query_max_response: optional string, containing an integer, in range 1 to 10
Configures the value of the "max-response" field in the multicast queries originated by the logical switch. Default: 1 second.

other_config : mcast_eth_src: optional string
Configures the source Ethernet address for queries originated by the logical switch.

other_config : mcast_ip4_src: optional string
Configures the source IPv4 address for queries originated by the logical switch.

other_config : mcast_ip6_src: optional string
Configures the source IPv6 address for queries originated by the logical switch.

Interconnection:

other_config : interconn-ts: optional string
The **name** of corresponding transit switch in **OVN_IC_Northbound** database. This kind of logical switch is created and controlled by **ovn−ic**.

Tunnel Key:

other_config : requested-tnl-key: optional string, containing an integer, in range 1 to 16,777,215
Configures the datapath tunnel key for the logical switch. Usually this is not needed because **ovn−northd** will assign an unique key for each datapath by itself. However, if it is configured, **ovn−northd** honors the configured value. The typical use case is for interconnection: the tunnel keys for transit switches need to be unique globally, so they are maintained in the global

OVN_IC_Southbound database, and **ovn−ic** simply syncs the value from **OVN_IC_Southbound** through this config.

Other options:

other_config : vlan-passthru: optional string, either **true** or **false**
Determines whether VLAN tagged incoming traffic should be allowed.

Common Columns:

external_ids: map of string-string pairs
See **External IDs** at the beginning of this document.

Logical_Switch_Port TABLE

A port within an L2 logical switch.

Summary:

Core Features:	
name	string (must be unique within table)
type	string
Options:	
options	map of string-string pairs
Options for router ports:	
options : router-port	optional string
options : nat-addresses	optional string
Options for localnet ports:	
options : network_name	optional string
Options for l2gateway ports:	
options : network_name	optional string
options : l2gateway-chassis	optional string
Options for vtep ports:	
options : vtep-physical-switch	optional string
options : vtep-logical-switch	optional string
VMI (or VIF) Options:	
options : requested-chassis	optional string
options : qos_max_rate	optional string
options : qos_burst	optional string
Virtual port Options:	
options : virtual-ip	optional string
options : virtual-parents	optional string
IP Multicast Snooping Options:	
options : mcast_flood	optional string, either **true** or **false**
options : mcast_flood_reports	optional string, either **true** or **false**
Containers:	
parent_name	optional string
tag_request	optional integer, in range 0 to 4,095
tag	optional integer, in range 1 to 4,095
Port State:	
up	optional boolean
enabled	optional boolean
Addressing:	
addresses	set of strings
dynamic_addresses	optional string
port_security	set of strings
DHCP:	
dhcpv4_options	optional weak reference to **DHCP_Options**
dhcpv6_options	optional weak reference to **DHCP_Options**
ha_chassis_group	optional **HA_Chassis_Group**
Naming:	
external_ids : neutron:port_name	optional string
Tunnel Key:	
options : requested-tnl-key	optional string, containing an integer, in range 1 to 32,767
Common Columns:	
external_ids	map of string-string pairs

Details:

Core Features:

name: string (must be unique within table)
The logical port name.

For entities (VMs or containers) that are spawned in the hypervisor, the name used here must match those used in the **external_ids:iface-id** in the **Open_vSwitch** database's **Interface** table, because hypervisors use **external_ids:iface-id** as a lookup key to identify the network interface of that entity.

For containers that share a VIF within a VM, the name can be any unique identifier. See **Containers**, below, for more information.

A logical switch port may not have the same name as a logical router port, but the database schema cannot enforce this.

type: string
Specify a type for this logical port. Logical ports can be used to model other types of connectivity into an OVN logical switch. The following types are defined:

(empty string)
A VM (or VIF) interface.

router A connection to a logical router. The value of **options:router-port** specifies the **name** of the **Logical_Router_Port** to which this logical switch port is connected.

localnet
A connection to a locally accessible network from **ovn−controller** instances that have a corresponding bridge mapping. A logical switch can have multiple **localnet** ports attached. This type is used to model direct connectivity to existing networks. In this case, each chassis should have a mapping for one of the physical networks only. Note: nothing said above implies that a chassis cannot be plugged to multiple physical networks as long as they belong to different switches.

localport
A connection to a local VIF. Traffic that arrives on a **localport** is never forwarded over a tunnel to another chassis. These ports are present on every chassis and have the same address in all of them. This is used to model connectivity to local services that run on every hypervisor.

l2gateway
A connection to a physical network.

vtep A port to a logical switch on a VTEP gateway.

external
Represents a logical port which is external and not having an OVS port in the integration bridge. **OVN** will never receive any traffic from this port or send any traffic to this port. **OVN** can support native services like DHCPv4/DHCPv6/DNS for this port. If **ha_chassis_group** is defined, **ovn−controller** running in the master chassis of the HA chassis group will bind this port to provide these native services. It is expected that this port belong to a bridged logical switch (with a **localnet** port).

It is recommended to use the same HA chassis group for all the external ports of a logical switch. Otherwise, the physical switch might see MAC flap issue when different chassis provide the native services. For example when supporting native DHCPv4 service, DHCPv4 server mac (configured in **options:server_mac** column in table **DHCP_Options**) originating from different ports can cause MAC flap issue. The MAC of the logical router IP(s) can also flap if the same HA chassis group is not set for all the external ports of a logical switch.

Below are some of the use cases where **external** ports can be used.

- VMs connected to SR-IOV nics - Traffic from these VMs by passes the kernel stack and local **ovn−controller** do not bind these ports and cannot serve the

native services.

- When CMS supports provisioning baremetal servers.

virtual Represents a logical port which does not have an OVS port in the integration bridge and has a virtual ip configured in the **options:virtual-ip** column. This virtual ip can move around between the logical ports configured in the **options:virtual-parents** column.

One of the use case where **virtual** ports can be used is.

- The **virtual ip** represents a load balancer vip and the **virtual parents** provide load balancer service in an active-standby setup with the active virtual parent owning the **virtual ip**.

remote A remote port is to model a port that resides remotely on another OVN, which is on the other side of a transit logical switch for OVN interconnection. This type of ports are created by **ovn−ic** instead of by CMS. Any change to the port will be automatically overwritten by **ovn−ic**.

Options:

options: map of string-string pairs
This column provides key/value settings specific to the logical port **type**. The type-specific options are described individually below.

Options for router ports:

These options apply when **type** is **router**.

options : router-port: optional string
Required. The **name** of the **Logical_Router_Port** to which this logical switch port is connected.

options : nat-addresses: optional string
This is used to send gratuitous ARPs for SNAT and DNAT IP addresses via the **localnet** port that is attached to the same logical switch as this type **router** port. This option is specified on a logical switch port that is connected to a gateway router, or a logical switch port that is connected to a distributed gateway port on a logical router.

This must take one of the following forms:

router Gratuitous ARPs will be sent for all SNAT and DNAT external IP addresses and for all load balancer IP addresses defined on the **options:router-port**'s logical router, using the **options:router-port**'s MAC address.

This form of **options:nat-addresses** is valid for logical switch ports where **options:router-port** is the name of a port on a gateway router, or the name of a distributed gateway port.

Supported only in OVN 2.8 and later. Earlier versions required NAT addresses to be manually synchronized.

Ethernet address followed by one or more IPv4 addresses
Example: **80:fa:5b:06:72:b7 158.36.44.22 158.36.44.24**. This would result in generation of gratuitous ARPs for IP addresses 158.36.44.22 and 158.36.44.24 with a MAC address of 80:fa:5b:06:72:b7.

This form of **options:nat-addresses** is only valid for logical switch ports where **options:router-port** is the name of a port on a gateway router.

Options for localnet ports:

These options apply when **type** is **localnet**.

options : network_name: optional string
Required. The name of the network to which the **localnet** port is connected. Each hypervisor, via **ovn−controller**, uses its local configuration to determine exactly how to connect to this locally

accessible network, if at all.

Options for l2gateway ports:

These options apply when **type** is **l2gateway**.

options : network_name: optional string
Required. The name of the network to which the **l2gateway** port is connected. The L2 gateway, via **ovn−controller**, uses its local configuration to determine exactly how to connect to this network.

options : l2gateway-chassis: optional string
Required. The chassis on which the **l2gateway** logical port should be bound to. **ovn−controller** running on the defined chassis will connect this logical port to the physical network.

Options for vtep ports:

These options apply when **type** is **vtep**.

options : vtep-physical-switch: optional string
Required. The name of the VTEP gateway.

options : vtep-logical-switch: optional string
Required. A logical switch name connected by the VTEP gateway.

VMI (or VIF) Options:

These options apply to logical ports with **type** having (empty string)

options : requested-chassis: optional string
If set, identifies a specific chassis (by name or hostname) that is allowed to bind this port. Using this option will prevent thrashing between two chassis trying to bind the same port during a live migration. It can also prevent similar thrashing due to a mis-configuration, if a port is accidentally created on more than one chassis.

options : qos_max_rate: optional string
If set, indicates the maximum rate for data sent from this interface, in bit/s. The traffic will be shaped according to this limit.

options : qos_burst: optional string
If set, indicates the maximum burst size for data sent from this interface, in bits.

Virtual port Options:

These options apply when **type** is **virtual**.

options : virtual-ip: optional string
This option represents the virtual IPv4 address.

options : virtual-parents: optional string
This options represents a set of logical port names (with in the same logical switch) which can own the **virtual ip** configured in the **options:virtual-ip**. All these virtual parents should add the **virtual ip** in the **port_security** if port security addressed are enabled.

IP Multicast Snooping Options:

These options apply when the port is part of a logical switch which has **other_config** :mcast_snoop set to **true**.

options : mcast_flood: optional string, either **true** or **false**
If set to **true**, multicast packets (except reports) are unconditionally forwarded to the specific port.

options : mcast_flood_reports: optional string, either **true** or **false**
If set to **true**, multicast reports are unconditionally forwarded to the specific port.

Containers:

When a large number of containers are nested within a VM, it may be too expensive to dedicate a VIF to

each container. OVN can use VLAN tags to support such cases. Each container is assigned a VLAN ID and each packet that passes between the hypervisor and the VM is tagged with the appropriate ID for the container. Such VLAN IDs never appear on a physical wire, even inside a tunnel, so they need not be unique except relative to a single VM on a hypervisor.

These columns are used for VIFs that represent nested containers using shared VIFs. For VMs and for containers that have dedicated VIFs, they are empty.

parent_name: optional string
The VM interface through which the nested container sends its network traffic. This must match the **name** column for some other **Logical_Switch_Port**.

tag_request: optional integer, in range 0 to 4,095
The VLAN tag in the network traffic associated with a container's network interface. The client can request **ovn−northd** to allocate a tag that is unique within the scope of a specific parent (specified in **parent_name**) by setting a value of **0** in this column. The allocated value is written by **ovn−northd** in the **tag** column. (Note that these tags are allocated and managed locally in **ovn−northd**, so they cannot be reconstructed in the event that the database is lost.) The client can also request a specific non-zero tag and **ovn−northd** will honor it and copy that value to the **tag** column.

When **type** is set to **localnet** or **l2gateway**, this can be set to indicate that the port represents a connection to a specific VLAN on a locally accessible network. The VLAN ID is used to match incoming traffic and is also added to outgoing traffic.

tag: optional integer, in range 1 to 4,095
The VLAN tag allocated by **ovn−northd** based on the contents of the **tag_request** column.

Port State:

up: optional boolean
This column is populated by **ovn−northd**, rather than by the CMS plugin as is most of this database. When a logical port is bound to a physical location in the OVN Southbound database **Binding** table, **ovn−northd** sets this column to **true**; otherwise, or if the port becomes unbound later, it sets it to **false**. If this column is empty, the port is not considered up. This allows the CMS to wait for a VM's (or container's) networking to become active before it allows the VM (or container) to start.

Logical ports of router type are an exception to this rule. They are considered to be always up, that is this column is always set to **true**.

enabled: optional boolean
This column is used to administratively set port state. If this column is empty or is set to **true**, the port is enabled. If this column is set to **false**, the port is disabled. A disabled port has all ingress and egress traffic dropped.

Addressing:

addresses: set of strings
Addresses owned by the logical port.

Each element in the set must take one of the following forms:

Ethernet address followed by zero or more IPv4 or IPv6 addresses (or both)
An Ethernet address defined is owned by the logical port. Like a physical Ethernet NIC, a logical port ordinarily has a single fixed Ethernet address.

When a OVN logical switch processes a unicast Ethernet frame whose destination MAC address is in a logical port's **addresses** column, it delivers it only to that port, as if a MAC learning process had learned that MAC address on the port.

If IPv4 or IPv6 address(es) (or both) are defined, it indicates that the logical port owns the given IP addresses.

If IPv4 address(es) are defined, the OVN logical switch uses this information to synthesize responses to ARP requests without traversing the physical network. The OVN logical router connected to the logical switch, if any, uses this information to avoid issuing ARP requests for logical switch ports.

Note that the order here is important. The Ethernet address must be listed before the IP address(es) if defined.

Examples:

80:fa:5b:06:72:b7
This indicates that the logical port owns the above mac address.

80:fa:5b:06:72:b7 10.0.0.4 20.0.0.4
This indicates that the logical port owns the mac address and two IPv4 addresses.

80:fa:5b:06:72:b7 fdaa:15f2:72cf:0:f816:3eff:fe20:3f41
This indicates that the logical port owns the mac address and 1 IPv6 address.

80:fa:5b:06:72:b7 10.0.0.4 fdaa:15f2:72cf:0:f816:3eff:fe20:3f41
This indicates that the logical port owns the mac address and 1 IPv4 address and 1 IPv6 address.

unknown
This indicates that the logical port has an unknown set of Ethernet addresses. When an OVN logical switch processes a unicast Ethernet frame whose destination MAC address is not in any logical port's **addresses** column, it delivers it to the port (or ports) whose **addresses** columns include **unknown**.

dynamic
Use **dynamic** to make **ovn−northd** generate a globally unique MAC address, choose an unused IPv4 address with the logical port's subnet (if **other_config:subnet** is set in the port's **Logical_Switch**), and generate an IPv6 address from the MAC address (if **other_config:ipv6_prefix** is set in the port's **Logical_Switch**) and store them in the port's **dynamic_addresses** column.

Only one element containing **dynamic** may appear in **addresses**.

dynamic *ip*
dynamic *ipv6*
dynamic *ip ipv6*
These act like **dynamic** alone but specify particular IPv4 or IPv6 addresses to use. OVN IPAM will still automatically allocate the other address if configured appropriately. Example: **dynamic 192.168.0.1 2001::1**.

mac **dynamic**
This acts like **dynamic** alone but specifies a particular MAC address to use. OVN IPAM will still automatically allocate IPv4 or IPv6 addresses, or both, if configured appropriately. Example: **80:fa:5b:06:72:b7 dynamic**

router Accepted only when **type** is **router**. This indicates that the Ethernet, IPv4, and IPv6 addresses for this logical switch port should be obtained from the connected logical router port, as specified by **router−port** in **options**.

The resulting addresses are used to populate the logical switch's destination lookup, and also for the logical switch to generate ARP and ND replies.

If the connected logical router port has a distributed gateway port specified and the logical router has rules specified in **nat** with **external_mac**, then those addresses are also used to populate the switch's destination lookup.

Supported only in OVN 2.7 and later. Earlier versions required router addresses to be manually synchronized.

dynamic_addresses: optional string

Addresses assigned to the logical port by **ovn−northd**, if **dynamic** is specified in **addresses**. Addresses will be of the same format as those that populate the **addresses** column. Note that dynamically assigned addresses are constructed and managed locally in ovn-northd, so they cannot be reconstructed in the event that the database is lost.

port_security: set of strings

This column controls the addresses from which the host attached to the logical port (“the host”) is allowed to send packets and to which it is allowed to receive packets. If this column is empty, all addresses are permitted.

Each element in the set must begin with one Ethernet address. This would restrict the host to sending packets from and receiving packets to the ethernet addresses defined in the logical port’s **port_security** column. It also restricts the inner source MAC addresses that the host may send in ARP and IPv6 Neighbor Discovery packets. The host is always allowed to receive packets to multicast and broadcast Ethernet addresses.

Each element in the set may additionally contain one or more IPv4 or IPv6 addresses (or both), with optional masks. If a mask is given, it must be a CIDR mask. In addition to the restrictions described for Ethernet addresses above, such an element restricts the IPv4 or IPv6 addresses from which the host may send and to which it may receive packets to the specified addresses. A masked address, if the host part is zero, indicates that the host is allowed to use any address in the subnet; if the host part is nonzero, the mask simply indicates the size of the subnet. In addition:

- If any IPv4 address is given, the host is also allowed to receive packets to the IPv4 local broadcast address 255.255.255.255 and to IPv4 multicast addresses (224.0.0.0/4). If an IPv4 address with a mask is given, the host is also allowed to receive packets to the broadcast address in that specified subnet.

  If any IPv4 address is given, the host is additionally restricted to sending ARP packets with the specified source IPv4 address. (RARP is not restricted.)

- If any IPv6 address is given, the host is also allowed to receive packets to IPv6 multicast addresses (ff00::/8).

  If any IPv6 address is given, the host is additionally restricted to sending IPv6 Neighbor Discovery Solicitation or Advertisement packets with the specified source address or, for solicitations, the unspecified address.

If an element includes an IPv4 address, but no IPv6 addresses, then IPv6 traffic is not allowed. If an element includes an IPv6 address, but no IPv4 address, then IPv4 and ARP traffic is not allowed.

This column uses the same lexical syntax as the **match** column in the OVN Southbound database’s **Pipeline** table. Multiple addresses within an element may be space or comma separated.

This column is provided as a convenience to cloud management systems, but all of the features that it implements can be implemented as ACLs using the **ACL** table.

Examples:

80:fa:5b:06:72:b7

The host may send traffic from and receive traffic to the specified MAC address, and to receive traffic to Ethernet multicast and broadcast addresses, but not otherwise. The host may not send ARP or IPv6 Neighbor Discovery packets with inner source Ethernet addresses other than the one specified.

80:fa:5b:06:72:b7 192.168.1.10/24

This adds further restrictions to the first example. The host may send IPv4 packets from or receive IPv4 packets to only 192.168.1.10, except that it may also receive IPv4 packets

to 192.168.1.255 (based on the subnet mask), 255.255.255.255, and any address in 224.0.0.0/4. The host may not send ARPs with a source Ethernet address other than 80:fa:5b:06:72:b7 or source IPv4 address other than 192.168.1.10. The host may not send or receive any IPv6 (including IPv6 Neighbor Discovery) traffic.

"80:fa:5b:12:42:ba", "80:fa:5b:06:72:b7 192.168.1.10/24"
The host may send traffic from and receive traffic to the specified MAC addresses, and to receive traffic to Ethernet multicast and broadcast addresses, but not otherwise. With MAC 80:fa:5b:12:42:ba, the host may send traffic from and receive traffic to any L3 address. With MAC 80:fa:5b:06:72:b7, the host may send IPv4 packets from or receive IPv4 packets to only 192.168.1.10, except that it may also receive IPv4 packets to 192.168.1.255 (based on the subnet mask), 255.255.255.255, and any address in 224.0.0.0/4. The host may not send or receive any IPv6 (including IPv6 Neighbor Discovery) traffic.

DHCP:

dhcpv4_options: optional weak reference to **DHCP_Options**
This column defines the DHCPv4 Options to be included by the **ovn−controller** when it replies to the DHCPv4 requests. Please see the **DHCP_Options** table.

dhcpv6_options: optional weak reference to **DHCP_Options**
This column defines the DHCPv6 Options to be included by the **ovn−controller** when it replies to the DHCPv6 requests. Please see the **DHCP_Options** table.

ha_chassis_group: optional **HA_Chassis_Group**
References a row in the OVN Northbound database's **HA_Chassis_Group** table. It indicates the HA chassis group to use if the **type** is set to **external**. If **type** is not **external**, this column is ignored.

Naming:

external_ids : neutron:port_name: optional string
This column gives an optional human-friendly name for the port. This name has no special meaning or purpose other than to provide convenience for human interaction with the northbound database.

Neutron copies this from its own port object's name. (Neutron ports do are not assigned human-friendly names by default, so it will often be empty.)

Tunnel Key:

options : requested-tnl-key: optional string, containing an integer, in range 1 to 32,767
Configures the port binding tunnel key for the port. Usually this is not needed because **ovn−northd** will assign an unique key for each port by itself. However, if it is configured, **ovn−northd** honors the configured value. The typical use case is for interconnection: the tunnel keys for ports on transit switches need to be unique globally, so they are maintained in the global **OVN_IC_Southbound** database, and **ovn−ic** simply syncs the value from **OVN_IC_Southbound** through this config.

Common Columns:

external_ids: map of string-string pairs
See **External IDs** at the beginning of this document.

The **ovn−northd** program copies all these pairs into the **external_ids** column of the **Port_Binding** table in **OVN_Southbound** database.

Forwarding_Group TABLE

Each row represents one forwarding group.

Summary:

name	string
vip	string
vmac	string
liveness	boolean
child_port	set of 1 or more strings
Common Columns:	
external_ids	map of string-string pairs

Details:

name: string
: A name for the forwarding group. This name has no special meaning or purpose other than to provide convenience for human interaction with the ovn-nb database.

vip: string
: The virtual IP address assigned to the forwarding group. It will respond with vmac when an ARP request is sent for vip.

vmac: string
: The virtual MAC address assigned to the forwarding group.

liveness: boolean
: If set to **true**, liveness is enabled for child ports otherwise it is disabled.

child_port: set of 1 or more strings
: List of child ports in the forwarding group.

Common Columns:

external_ids: map of string-string pairs
: See **External IDs** at the beginning of this document.

Address_Set TABLE

Each row in this table represents a named set of addresses. An address set may contain Ethernet, IPv4, or IPv6 addresses with optional bitwise or CIDR masks. Address set may ultimately be used in ACLs to compare against fields such as **ip4.src** or **ip6.src**. A single address set must contain addresses of the same type. As an example, the following would create an address set with three IP addresses:

ovn−nbctl create Address_Set name=set1 addresses='10.0.0.1 10.0.0.2 10.0.0.3'

Address sets may be used in the **match** column of the **ACL** table. For syntax information, see the details of the expression language used for the **match** column in the **Logical_Flow** table of the **OVN_Southbound** database.

Summary:

name	string (must be unique within table)
addresses	set of strings
Common Columns:	
external_ids	map of string-string pairs

Details:

name: string (must be unique within table)

A name for the address set. Names are ASCII and must match **[a−zA−Z_.][a−zA−Z_.0−9]***.

addresses: set of strings

The set of addresses in string form.

Common Columns:

external_ids: map of string-string pairs

See **External IDs** at the beginning of this document.

Port_Group TABLE

Each row in this table represents a named group of logical switch ports.

Port groups may be used in the **match** column of the **ACL** table. For syntax information, see the details of the expression language used for the **match** column in the **Logical_Flow** table of the **OVN_Southbound** database.

For each port group, there are two address sets generated to the **Address_Set** table of the **OVN_Southbound** database, containing the IP addresses of the group of ports, one for IPv4, and the other for IPv6, with **name** being the **name** of the **Port_Group** followed by a suffix **_ip4** for IPv4 and **_ip6** for IPv6. The generated address sets can be used in the same way as regular address sets in the **match** column of the **ACL** table. For syntax information, see the details of the expression language used for the **match** column in the **Logical_Flow** table of the **OVN_Southbound** database.

Summary:

name	string (must be unique within table)
ports	set of weak reference to **Logical_Switch_Port**s
acls	set of **ACL**s
Common Columns:	
external_ids	map of string-string pairs

Details:

name: string (must be unique within table)

A name for the port group. Names are ASCII and must match **[a−zA−Z_.][a−zA−Z_.0−9]***.

ports: set of weak reference to **Logical_Switch_Port**s

The logical switch ports belonging to the group in uuids.

acls: set of **ACL**s

Access control rules that apply to the port group. Applying an ACL to a port group has the same effect as applying the ACL to all logical lswitches that the ports of the port group belong to.

Common Columns:

external_ids: map of string-string pairs

See **External IDs** at the beginning of this document.

Load_Balancer TABLE

Each row represents one load balancer.

Summary:

name	string
vips	map of string-string pairs
protocol	optional string, one of **sctp**, **tcp**, or **udp**
Health Checks:	
health_check	set of **Load_Balancer_Health_Check**s
ip_port_mappings	map of string-string pairs
selection_fields	set of strings, one of **eth_dst**, **eth_src**, **ip_dst**, **ip_src**, **tp_dst**, or **tp_src**
Common Columns:	
external_ids	map of string-string pairs
Load_Balancer options:	
options : reject	optional string, either **true** or **false**
options : hairpin_snat_ip	optional string

Details:

name: string

A name for the load balancer. This name has no special meaning or purpose other than to provide convenience for human interaction with the ovn-nb database.

vips: map of string-string pairs

A map of virtual IP addresses (and an optional port number with **:** as a separator) associated with this load balancer and their corresponding endpoint IP addresses (and optional port numbers with **:** as separators) separated by commas. If the destination IP address (and port number) of a packet leaving a container or a VM matches the virtual IP address (and port number) provided here as a key, then OVN will statefully replace the destination IP address by one of the provided IP address (and port number) in this map as a value. IPv4 and IPv6 addresses are supported for load balancing; however a VIP of one address family may not be mapped to a destination IP address of a different family. If specifying an IPv6 address with a port, the address portion must be enclosed in square brackets. Examples for keys are "192.168.1.4" and "[fd0f::1]:8800". Examples for value are "10.0.0.1, 10.0.0.2" and "20.0.0.10:8800, 20.0.0.11:8800".

When the **Load_Balancer** is added to the **logical_switch**, the VIP has to be in a different subnet than the one used for the **logical_switch**. Since VIP is in a different subnet, you should connect your logical switch to either a OVN logical router or a real router (this is because the client can now send a packet with VIP as the destination IP address and router's mac address as the destination MAC address).

protocol: optional string, one of **sctp**, **tcp**, or **udp**

Valid protocols are **tcp**, **udp**, or **sctp**. This column is useful when a port number is provided as part of the **vips** column. If this column is empty and a port number is provided as part of **vips** column, OVN assumes the protocol to be **tcp**.

Health Checks:

OVN supports health checks for load balancer endpoints, for IPv4 load balancers only. When health checks are enabled, the load balancer uses only healthy endpoints.

Suppose that **vips** contains a key-value pair **10.0.0.10:80=10.0.0.4:8080,20.0.0.4:8080**. To enable health checks for this virtual's endpoints, add two key-value pairs to **ip_port_mappings**, with keys **10.0.0.4** and **20.0.0.4**, and add to **health_check** a reference to a **Load_Balancer_Health_Check** row whose **vip** is set to **10.0.0.10**.

health_check: set of **Load_Balancer_Health_Check**s

Load balancer health checks associated with this load balancer.

ip_port_mappings: map of string-string pairs
Maps from endpoint IP to a colon-separated pair of logical port name and source IP, e.g. *port_name***:***sourc_ip*. Health checks are sent to this port with the specified source IP.

For example, in the example above, IP to port mappings might be defined as **10.0.0.4=sw0−p1:10.0.0.2** and **20.0.0.4=sw1−p1:20.0.0.2**, if the values given were suitable ports and IP addresses.

selection_fields: set of strings, one of **eth_dst**, **eth_src**, **ip_dst**, **ip_src**, **tp_dst**, or **tp_src**
OVN native load balancers are supported using the OpenFlow groups of type **select**. OVS supports two selection methods: **dp_hash** and **hash (with optional fields specified)** in selecting the buckets of a group. Please see the OVS documentation (man ovs-ofctl) for more details on the selection methods. Each endpoint IP (and port if set) is mapped to a bucket in the group flow.

CMS can choose the **hash** selection method by setting the selection fields in this column. **ovs−vswitchd** uses the specified fields in generating the hash.

dp_hash selection method uses the assistance of datapath to calculate the hash and it is expected to be faster than **hash** selection method. So CMS should take this into consideration before using the **hash** method. Please consult the OVS documentation and OVS sources for the implementation details.

Common Columns:

external_ids: map of string-string pairs
See **External IDs** at the beginning of this document.

Load_Balancer options:

options : reject: optional string, either **true** or **false**
If the load balancer is created with **−−reject** option and it has no active backends, a TCP reset segment (for tcp) or an ICMP port unreachable packet (for all other kind of traffic) will be sent whenever an incoming packet is received for this load-balancer. Please note using **−−reject** option will disable empty_lb SB controller event for this load balancer.

options : hairpin_snat_ip: optional string
IP to be used as source IP for packets that have been hair-pinned after load balancing. The default behavior when the option is not set is to use the load balancer VIP as source IP. This option may have exactly one IPv4 and/or one IPv6 address on it, separated by a space character.

Load_Balancer_Health_Check TABLE

Each row represents one load balancer health check. Health checks are supported for IPv4 load balancers only.

Summary:

vip	string
Health check options:	
options : interval	optional string, containing an integer
options : timeout	optional string, containing an integer
options : success_count	optional string, containing an integer
options : failure_count	optional string, containing an integer
Common Columns:	
external_ids	map of string-string pairs

Details:

vip: string

vip whose endpoints should be monitored for health check.

Health check options:

options : interval: optional string, containing an integer

The interval, in seconds, between health checks.

options : timeout: optional string, containing an integer

The time, in seconds, after which a health check times out.

options : success_count: optional string, containing an integer

The number of successful checks after which the endpoint is considered online.

options : failure_count: optional string, containing an integer

The number of failure checks after which the endpoint is considered offline.

Common Columns:

external_ids: map of string-string pairs

See **External IDs** at the beginning of this document.

ACL TABLE

Each row in this table represents one ACL rule for a logical switch or a port group that points to it through its **acls** column. The **action** column for the highest-**priority** matching row in this table determines a packet's treatment. If no row matches, packets are allowed by default. (Default-deny treatment is possible: add a rule with **priority** 0, **1** as **match**, and **deny** as **action**.)

Summary:

priority	integer, in range 0 to 32,767
direction	string, either **from−lport** or **to−lport**
match	string
action	string, one of **allow−related**, **allow**, **drop**, or **reject**
Logging:	
log	boolean
name	optional string, at most 63 characters long
severity	optional string, one of **alert**, **debug**, **info**, **notice**, or **warning**
meter	optional string
Common Columns:	
external_ids	map of string-string pairs

Details:

priority: integer, in range 0 to 32,767

The ACL rule's priority. Rules with numerically higher priority take precedence over those with lower. If two ACL rules with the same priority both match, then the one actually applied to a packet is undefined.

Return traffic from an **allow−related** flow is always allowed and cannot be changed through an ACL.

direction: string, either **from−lport** or **to−lport**

Direction of the traffic to which this rule should apply:

- **from−lport**: Used to implement filters on traffic arriving from a logical port. These rules are applied to the logical switch's ingress pipeline.
- **to−lport**: Used to implement filters on traffic forwarded to a logical port. These rules are applied to the logical switch's egress pipeline.

match: string

The packets that the ACL should match, in the same expression language used for the **match** column in the OVN Southbound database's **Logical_Flow** table. The **outport** logical port is only available in the **to−lport** direction (the **inport** is available in both directions).

By default all traffic is allowed. When writing a more restrictive policy, it is important to remember to allow flows such as ARP and IPv6 neighbor discovery packets.

Note that you can not create an ACL matching on a port with type=router or type=localnet.

action: string, one of **allow−related**, **allow**, **drop**, or **reject**

The action to take when the ACL rule matches:

- **allow**: Forward the packet.
- **allow−related**: Forward the packet and related traffic (e.g. inbound replies to an outbound connection).
- **drop**: Silently drop the packet.
- **reject**: Drop the packet, replying with a RST for TCP or ICMPv4/ICMPv6 unreachable message for other IPv4/IPv6-based protocols.

Logging:

These columns control whether and how OVN logs packets that match an ACL.

log: boolean
: If set to **true**, packets that match the ACL will trigger a log message on the transport node or nodes that perform ACL processing. Logging may be combined with any **action**.

  If set to **false**, the remaining columns in this group have no significance.

name: optional string, at most 63 characters long
: This name, if it is provided, is included in log records. It provides the administrator and the cloud management system a way to associate a log record with a particular ACL.

severity: optional string, one of **alert**, **debug**, **info**, **notice**, or **warning**
: The severity of the ACL. The severity levels match those of syslog, in decreasing level of severity: **alert**, **warning**, **notice**, **info**, or **debug**. When the column is empty, the default is **info**.

meter: optional string
: The name of a meter to rate-limit log messages for the ACL. The string must match the **name** column of a row in the **Meter** table. By default, log messages are not rate-limited. In order to ensure that the same **Meter** rate limits multiple ACL logs separately, set the **fair** column.

Common Columns:

external_ids: map of string-string pairs
: See **External IDs** at the beginning of this document.

Logical_Router TABLE

Each row represents one L3 logical router.

Summary:

ports	set of **Logical_Router_Port**s
static_routes	set of **Logical_Router_Static_Route**s
policies	set of **Logical_Router_Policy**s
enabled	optional boolean
nat	set of **NAT**s
load_balancer	set of weak reference to **Load_Balancer**s
Naming:	
name	string
external_ids : neutron:router_name	optional string
Options:	
options : chassis	optional string
options : dnat_force_snat_ip	optional string
options : lb_force_snat_ip	optional string
options : mcast_relay	optional string, either **true** or **false**
options : dynamic_neigh_routers	optional string, either **true** or **false**
options : always_learn_from_arp_request	optional string, either **true** or **false**
options : requested-tnl-key	optional string, containing an integer, in range 1 to 16,777,215
options : snat-ct-zone	optional string, containing an integer, in range 0 to 65,535
Common Columns:	
external_ids	map of string-string pairs

Details:

ports: set of **Logical_Router_Port**s

The router's ports.

static_routes: set of **Logical_Router_Static_Route**s

Zero or more static routes for the router.

policies: set of **Logical_Router_Policy**s

Zero or more routing policies for the router.

enabled: optional boolean

This column is used to administratively set router state. If this column is empty or is set to **true**, the router is enabled. If this column is set to **false**, the router is disabled. A disabled router has all ingress and egress traffic dropped.

nat: set of **NAT**s

One or more NAT rules for the router. NAT rules only work on Gateway routers, and on distributed routers with logical gateway ports.

load_balancer: set of weak reference to **Load_Balancer**s

Load balance a virtual ip address to a set of logical port ip addresses. Load balancer rules only work on the Gateway routers or routers with distributed gateway ports.

Naming:

These columns provide names for the logical router. From OVN's perspective, these names have no special meaning or purpose other than to provide convenience for human interaction with the northbound database. There is no requirement for the name to be unique. (For a unique identifier for a logical router, use its row UUID.)

(Originally, **name** was intended to serve the purpose of a human-friendly name, but the Neutron integration used it to uniquely identify its own router object, in the format **neutron–***uuid*. Later on, Neutron started propagating the friendly name of a router as **external_ids:neutron:router_name**. Perhaps this can be

cleaned up someday.)

name: string
A name for the logical router.

external_ids : neutron:router_name: optional string
Another name for the logical router.

Options:

Additional options for the logical router.

options : chassis: optional string
If set, indicates that the logical router in question is a Gateway router (which is centralized) and resides in the set chassis. The same value is also used by **ovn−controller** to uniquely identify the chassis in the OVN deployment and comes from **external_ids:system−id** in the **Open_vSwitch** table of Open_vSwitch database.

The Gateway router can only be connected to a distributed router via a switch if SNAT and DNAT are to be configured in the Gateway router.

options : dnat_force_snat_ip: optional string
If set, indicates a set of IP addresses to use to force SNAT a packet that has already been DNATed in the gateway router. When multiple gateway routers are configured, a packet can potentially enter any of the gateway router, get DNATted and eventually reach the logical switch port. For the return traffic to go back to the same gateway router (for unDNATing), the packet needs a SNAT in the first place. This can be achieved by setting the above option with a gateway specific set of IP addresses. This option may have exactly one IPv4 and/or one IPv6 address on it, separated by a a space.

options : lb_force_snat_ip: optional string
If set, this option can take two possible type of values. Either a set of IP addresses or the string value - **router_ip**.

If a set of IP addresses are configured, it indicates to use to force SNAT a packet that has already been load-balanced in the gateway router. When multiple gateway routers are configured, a packet can potentially enter any of the gateway routers, get DNATted as part of the load-balancing and eventually reach the logical switch port. For the return traffic to go back to the same gateway router (for unDNATing), the packet needs a SNAT in the first place. This can be achieved by setting the above option with a gateway specific set of IP addresses. This option may have exactly one IPv4 and/or one IPv6 address on it, separated by a space character.

If it is configured with the value **router_ip**, then the load balanced packet is SNATed with the IP of router port (attached to the gateway router) selected as the destination after taking the routing decision.

options : mcast_relay: optional string, either **true** or **false**
Enables/disables IP multicast relay between logical switches connected to the logical router. Default: False.

options : dynamic_neigh_routers: optional string, either **true** or **false**
If set to **true**, the router will resolve neighbor routers' MAC addresses only by dynamic ARP/ND, instead of prepopulating static mappings for all neighbor routers in the ARP/ND Resolution stage. This reduces number of flows, but requires ARP/ND messages to resolve the IP-MAC bindings when needed. It is **false** by default. It is recommended to set to **true** when a large number of logical routers are connected to the same logical switch but most of them never need to send traffic between each other.

options : always_learn_from_arp_request: optional string, either **true** or **false**
This option controls the behavior when handling IPv4 ARP requests or IPv6 ND-NS packets - whether a dynamic neighbor (MAC binding) entry is added/updated.

true - Always learn the MAC-IP binding, and add/update the MAC binding entry.

false - If there is a MAC binding for that IP and the MAC is different, or, if TPA of ARP request belongs to any router port on this router, then update/add that MAC-IP binding. Otherwise, don't update/add entries.

It is **true** by default. It is recommended to set to **false** when a large number of logical routers are connected to the same logical switch but most of them never need to send traffic between each other, to reduce the size of the MAC binding table.

options : requested-tnl-key: optional string, containing an integer, in range 1 to 16,777,215
Configures the datapath tunnel key for the logical router. This is not needed because **ovn−northd** will assign an unique key for each datapath by itself. However, if it is configured, **ovn−northd** honors the configured value.

options : snat-ct-zone: optional string, containing an integer, in range 0 to 65,535
Use the requested conntrack zone for SNAT with this router. This can be useful if egress traffic from the host running OVN comes from both OVN and other sources. This way, OVN and the other sources can make use of the same conntrack zone.

Common Columns:

external_ids: map of string-string pairs
See **External IDs** at the beginning of this document.

QoS TABLE

Each row in this table represents one QoS rule for a logical switch that points to it through its **qos_rules** column. Two types of QoS are supported: DSCP marking and metering. A **match** with the highest-**priority** will have QoS applied to it. If the **action** column is specified, then matching packets will have DSCP marking applied. If the **bandwdith** column is specified, then matching packets will have metering applied. **action** and **bandwdith** are not exclusive, so both marking and metering by defined for the same QoS entry. If no row matches, packets will not have any QoS applied.

Summary:

priority	integer, in range 0 to 32,767
direction	string, either **from−lport** or **to−lport**
match	string
action	map of string-integer pairs, key must be **dscp**, value in range 0 to 63
bandwidth	map of string-integer pairs, key either **burst** or **rate**, value in range 1 to 4,294,967,295
external_ids	map of string-string pairs

Details:

priority: integer, in range 0 to 32,767

The QoS rule's priority. Rules with numerically higher priority take precedence over those with lower. If two QoS rules with the same priority both match, then the one actually applied to a packet is undefined.

direction: string, either **from−lport** or **to−lport**

The value of this field is similar to **ACL** column in the OVN Northbound database's **ACL** table.

match: string

The packets that the QoS rules should match, in the same expression language used for the **match** column in the OVN Southbound database's **Logical_Flow** table. The **outport** logical port is only available in the **to−lport** direction (the **inport** is available in both directions).

action: map of string-integer pairs, key must be **dscp**, value in range 0 to 63

When specified, matching flows will have DSCP marking applied.

- **dscp**: The value of this action should be in the range of 0 to 63 (inclusive).

bandwidth: map of string-integer pairs, key either **burst** or **rate**, value in range 1 to 4,294,967,295

When specified, matching packets will have bandwidth metering applied. Traffic over the limit will be dropped.

- **rate**: The value of rate limit in kbps.
- **burst**: The value of burst rate limit in kilobits. This is optional and needs to specify the **rate**.

external_ids: map of string-string pairs

See **External IDs** at the beginning of this document.

Meter TABLE

Each row in this table represents a meter that can be used for QoS or rate-limiting.

Summary:

name	string (must be unique within table)
unit	string, either **kbps** or **pktps**
bands	set of 1 or more **Meter_Band**s
fair	optional boolean
external_ids	map of string-string pairs

Details:

name: string (must be unique within table)

A name for this meter.

Names that begin with "__" (two underscores) are reserved for OVN internal use and should not be added manually.

unit: string, either **kbps** or **pktps**

The unit for **rate** and **burst_rate** parameters in the **bands** entry. **kbps** specifies kilobits per second, and **pktps** specifies packets per second.

bands: set of 1 or more **Meter_Band**s

The bands associated with this meter. Each band specifies a rate above which the band is to take the action **action**. If multiple bands' rates are exceeded, then the band with the highest rate among the exceeded bands is selected.

fair: optional boolean

This column is used to further describe the desired behavior of the meter when there are multiple references to it. If this column is empty or is set to **false**, the rate will be shared across all rows that refer to the same Meter **name**. Conversely, when this column is set to **true**, each user of the same Meter will be rate-limited on its own.

external_ids: map of string-string pairs

See **External IDs** at the beginning of this document.

Meter_Band TABLE

Each row in this table represents a meter band which specifies the rate above which the configured action should be applied. These bands are referenced by the **bands** column in the **Meter** table.

Summary:

action	string, must be **drop**
rate	integer, in range 1 to 4,294,967,295
burst_size	integer, in range 0 to 4,294,967,295
external_ids	map of string-string pairs

Details:

action: string, must be **drop**

The action to execute when this band matches. The only supported action is **drop**.

rate: integer, in range 1 to 4,294,967,295

The rate limit for this band, in kilobits per second or bits per second, depending on whether the parent **Meter** entry's **unit** column specified **kbps** or **pktps**.

burst_size: integer, in range 0 to 4,294,967,295

The maximum burst allowed for the band in kilobits or packets, depending on whether **kbps** or **pktps** was selected in the parent **Meter** entry's **unit** column. If the size is zero, the switch is free to select some reasonable value depending on its configuration.

external_ids: map of string-string pairs

See **External IDs** at the beginning of this document.

Logical_Router_Port TABLE

A port within an L3 logical router.

Exactly one **Logical_Router** row must reference a given logical router port.

Summary:

name	string (must be unique within table)
networks	set of 1 or more strings
mac	string
enabled	optional boolean
Distributed Gateway Ports:	
ha_chassis_group	optional **HA_Chassis_Group**
gateway_chassis	set of **Gateway_Chassis**es
Options for Physical VLAN MTU Issues:	
options : reside-on-redirect-chassis	optional string, either **true** or **false**
options : redirect-type	optional string, either **bridged** or **overlay**
ipv6_prefix	set of strings
ipv6_ra_configs:	
ipv6_ra_configs : address_mode	optional string
ipv6_ra_configs : router_preference	optional string
ipv6_ra_configs : route_info	optional string
ipv6_ra_configs : mtu	optional string
ipv6_ra_configs : send_periodic	optional string
ipv6_ra_configs : max_interval	optional string
ipv6_ra_configs : min_interval	optional string
ipv6_ra_configs : rdnss	optional string
ipv6_ra_configs : dnssl	optional string
Options:	
options : mcast_flood	optional string, either **true** or **false**
options : requested-tnl-key	optional string, containing an integer, in range 1 to 32,767
options : prefix_delegation	optional string, either **true** or **false**
options : prefix	optional string, either **true** or **false**
Attachment:	
peer	optional string
Common Columns:	
external_ids	map of string-string pairs

Details:

name: string (must be unique within table)

A name for the logical router port.

In addition to provide convenience for human interaction with the northbound database, this column is used as reference by its patch port in **Logical_Switch_Port** or another logical router port in **Logical_Router_Port**.

A logical router port may not have the same name as a logical switch port, but the database schema cannot enforce this.

networks: set of 1 or more strings

The IP addresses and netmasks of the router. For example, **192.168.0.1/24** indicates that the router's IP address is 192.168.0.1 and that packets destined to 192.168.0.*x* should be routed to this port.

A logical router port always adds a link-local IPv6 address (fe80::/64) automatically generated from the interface's MAC address using the modified EUI−64 format.

mac: string
The Ethernet address that belongs to this router port.

enabled: optional boolean
This column is used to administratively set port state. If this column is empty or is set to **true**, the port is enabled. If this column is set to **false**, the port is disabled. A disabled port has all ingress and egress traffic dropped.

Distributed Gateway Ports:

Gateways, as documented under **Gateways** in the OVN architecture guide, provide limited connectivity between logical networks and physical ones. OVN support multiple kinds of gateways. The **Logical_Router_Port** table can be used two different ways to configure *distributed gateway ports*, which are one kind of gateway. These two forms of configuration exist for historical reasons. Both of them produce the same kind of OVN southbound records and the same behavior in practice.

If either of these are set, this logical router port represents a distributed gateway port that connects this router to a logical switch with a **localnet** port or a connection to another OVN deployment. There may be at most one such logical router port on each logical router.

The preferred way to configure a gateway is **ha_chassis_group**, but **gateway_chassis** is also supported for backward compatibility. Only one of these should be set at a time on a given LRP, since they configure the same features.

Even when a gateway is configured, the logical router port still effectively resides on each chassis. However, due to the implications of the use of L2 learning in the physical network, as well as the need to support advanced features such as one-to-many NAT (aka IP masquerading), a subset of the logical router processing is handled in a centralized manner on the gateway chassis.

When more than one gateway chassis is specified, OVN only uses one at a time. OVN can rely on OVS BFD implementation to monitor gateway connectivity, preferring the highest-priority gateway that is online. Priorities are specified in the **priority** column of **Gateway_Chassis** or **HA_Chassis**.

ovn−northd programs the **external_mac** rules specified in the LRP's LR into the peer logical switch's destination lookup on the chassis where the **logical_port** resides. In addition, the logical router's MAC address is automatically programmed in the peer logical switch's destination lookup flow on the gateway chasssis. If it is desired to generate gratuitous ARPs for NAT addresses, then set the peer LSP's **options:nat-addresses** to **router**.

OVN 20.03 and earlier supported a third way to configure distributed gateway ports using **options:redirect−chassis** to specify the gateway chassis. This method is no longer supported. Any remaining users should switch to one of the newer methods instead. A **gateway_chassis** may be easily configured from the command line, e.g. **ovn−nbctl lrp−set−gateway−chassis** *lrp chassis*.

ha_chassis_group: optional **HA_Chassis_Group**
Designates an **HA_Chassis_Group** to provide gateway high availability.

gateway_chassis: set of **Gateway_Chassis**es
Designates one or more **Gateway_Chassis** for the logical router port.

Options for Physical VLAN MTU Issues:

MTU issues arise in mixing tunnels with logical networks that are bridged to a physical VLAN. For an explanation of the MTU issues, see **Physical VLAN MTU Issues** in the OVN architecture document. The following options, which are alternatives, provide solutions. Both of them cause packets to be sent over **localnet** instead of tunnels, but they differ in whether some or all packets are sent this way. The most prominent tradeoff between these options is that **reside−on−redirect−chassis** is easier to configure and that **redirect−type** performs better for east-west traffic.

options : reside-on-redirect-chassis: optional string, either **true** or **false**
If set to **true**, this option forces all traffic across the logical router port to pass through the gateway chassis using a hop across a **localnet** port. This changes behavior in two ways:

- Without this option, east-west traffic passes directly between source and destination chassis (or even within a single chassis, for co-located VMs). With this option, all east-west traffic passes through the gateway chassis.
- Without this option, traffic between the gateway chassis and other chassis is encapsulated in tunnels. With this option, traffic passes over a **localnet** interface.

This option may usefully be set only on logical router ports that connect a distributed logical router to a logical switch with VIFs. It should not be set on a distributed gateway port.

OVN honors this option only if the logical router has a distributed gateway port and if the LRP's peer switch has a **localnet** port.

options : redirect-type: optional string, either **bridged** or **overlay**
If set to **bridged** on a distributed gateway port, this option causes OVN to redirect packets to the gateway chassis over a **localnet** port instead of a tunnel. The relevant chassis must share a **localnet** port.

This feature requires the administrator or the CMS to configure each participating chassis with a unique Ethernet address for the logical router by setting **ovn−chassis−mac−mappings** in the Open vSwitch database, for use by **ovn−controller**.

Setting this option to **overlay** or leaving it unset has no effect. This option may usefully be set only on a distributed gateway port. It is otherwise ignored.

ipv6_prefix: set of strings
This column contains IPv6 prefix obtained by prefix delegation router according to RFC 3633

ipv6_ra_configs:

This column defines the IPv6 ND RA address mode and ND MTU Option to be included by **ovn−controller** when it replies to the IPv6 Router solicitation requests.

ipv6_ra_configs : address_mode: optional string
The address mode to be used for IPv6 address configuration. The supported values are:

- **slaac**: Address configuration using Router Advertisement (RA) packet. The IPv6 prefixes defined in the **Logical_Router_Port** table's **networks** column will be included in the RA's ICMPv6 option - Prefix information.
- **dhcpv6_stateful**: Address configuration using DHCPv6.
- **dhcpv6_stateless**: Address configuration using Router Advertisement (RA) packet. Other IPv6 options are provided by DHCPv6.

ipv6_ra_configs : router_preference: optional string
Default Router Preference (PRF) indicates whether to prefer this router over other default routers (RFC 4191). Possible values are:

- HIGH: mapped to 0x01 in RA PRF field
- MEDIUM: mapped to 0x00 in RA PRF field
- LOW: mapped to 0x11 in RA PRF field

ipv6_ra_configs : route_info: optional string
Route Info is used to configure Route Info Option sent in Router Advertisment according to RFC 4191. Route Info is a comma separated string where each field provides PRF and prefix for a given route (e.g: HIGH-aef1::11/48,LOW-aef2::11/96) Possible PRF values are:

- HIGH: mapped to 0x01 in RA PRF field
- MEDIUM: mapped to 0x00 in RA PRF field
- LOW: mapped to 0x11 in RA PRF field

ipv6_ra_configs : mtu: optional string
The recommended MTU for the link. Default is 0, which means no MTU Option will be included in RA packet replied by ovn-controller. Per RFC 2460, the mtu value is recommended no less than 1280, so any mtu value less than 1280 will be considered as no MTU Option.

ipv6_ra_configs : send_periodic: optional string
If set to true, then this router interface will send router advertisements periodically. The default is false.

ipv6_ra_configs : max_interval: optional string
The maximum number of seconds to wait between sending periodic router advertisements. This option has no effect if **ipv6_ra_configs:send_periodic** is false. The default is 600.

ipv6_ra_configs : min_interval: optional string
The minimum number of seconds to wait between sending periodic router advertisements. This option has no effect if **ipv6_ra_configs:send_periodic** is false. The default is one-third of **ipv6_ra_configs:max_interval**, i.e. 200 seconds if that key is unset.

ipv6_ra_configs : rdnss: optional string
IPv6 address of RDNSS server announced in RA packets. At the moment OVN supports just one RDNSS server.

ipv6_ra_configs : dnssl: optional string
DNS Search List announced in RA packets. Multiple DNS Search List must be 'comma' separated (e.g. "a.b.c, d.e.f")

Options:

Additional options for the logical router port.

options : mcast_flood: optional string, either **true** or **false**
If set to **true**, multicast traffic (including reports) are unconditionally forwarded to the specific port.

This option applies when the port is part of a logical router which has **options**:mcast_relay set to **true**.

options : requested-tnl-key: optional string, containing an integer, in range 1 to 32,767
Configures the port binding tunnel key for the port. Usually this is not needed because **ovn−northd** will assign an unique key for each port by itself. However, if it is configured, **ovn−northd** honors the configured value.

options : prefix_delegation: optional string, either **true** or **false**
If set to **true**, enable IPv6 prefix delegation state machine on this logical router port (RFC3633). IPv6 prefix delegation is available just on a gateway router or on a gateway router port.

options : prefix: optional string, either **true** or **false**
If set to **true**, this interface will receive an IPv6 prefix according to RFC3663

Attachment:

A given router port serves one of two purposes:

- To attach a logical switch to a logical router. A logical router port of this type is referenced by exactly one **Logical_Switch_Port** of type **router**. The value of **name** is set as **router−port** in column **options** of **Logical_Switch_Port**. In this case **peer** column is empty.
- To connect one logical router to another. This requires a pair of logical router ports, each connected to a different router. Each router port in the pair specifies the other in its **peer** column. No **Logical_Switch** refers to the router port.

peer: optional string
For a router port used to connect two logical routers, this identifies the other router port in the pair by **name**.

For a router port attached to a logical switch, this column is empty.

Common Columns:

external_ids: map of string-string pairs
See **External IDs** at the beginning of this document.

Logical_Router_Static_Route TABLE

Each record represents a static route.

When multiple routes match a packet, the longest-prefix match is chosen. For a given prefix length, a **dst−ip** route is preferred over a **src−ip** route.

When there are ECMP routes, i.e. multiple routes with same prefix and policy, one of them will be selected based on the 5-tuple hashing of the packet header.

Summary:

ip_prefix	string
policy	optional string, either **dst−ip** or **src−ip**
nexthop	string
output_port	optional string
bfd	optional weak reference to **BFD**
external_ids : ic-learned-route	optional string
Common Columns:	
external_ids	map of string-string pairs
Common options:	
options	map of string-string pairs
options : ecmp_symmetric_reply	optional string

Details:

ip_prefix: string
IP prefix of this route (e.g. 192.168.100.0/24).

policy: optional string, either **dst−ip** or **src−ip**
If it is specified, this setting describes the policy used to make routing decisions. This setting must be one of the following strings:

- **src−ip**: This policy sends the packet to the **nexthop** when the packet's source IP address matches **ip_prefix**.
- **dst−ip**: This policy sends the packet to the **nexthop** when the packet's destination IP address matches **ip_prefix**.

If not specified, the default is **dst−ip**.

nexthop: string
Nexthop IP address for this route. Nexthop IP address should be the IP address of a connected router port or the IP address of a logical port.

output_port: optional string
The name of the **Logical_Router_Port** via which the packet needs to be sent out. This is optional and when not specified, OVN will automatically figure this out based on the **nexthop**. When this is specified and there are multiple IP addresses on the router port and none of them are in the same subnet of **nexthop**, OVN chooses the first IP address as the one via which the **nexthop** is reachable.

bfd: optional weak reference to **BFD**
Reference to **BFD** row if the route has associated a BFD session

external_ids : ic-learned-route: optional string
ovn−ic populates this key if the route is learned from the global **OVN_IC_Southbound** database. In this case the value will be set to the uuid of the row in **Route** table of the **OVN_IC_Southbound** database.

Common Columns:

external_ids: map of string-string pairs
See **External IDs** at the beginning of this document.

Common options:

options: map of string-string pairs
This column provides general key/value settings. The supported options are described individually below.

options : ecmp_symmetric_reply: optional string
It true, then new traffic that arrives over this route will have its reply traffic bypass ECMP route selection and will be sent out this route instead. Note that this option overrides any rules set in the **Logical_Router_policy** table. This option only works on gateway routers (routers that have **options:chassis** set).

Logical_Router_Policy TABLE

Each row in this table represents one routing policy for a logical router that points to it through its **policies** column. The **action** column for the highest-**priority** matching row in this table determines a packet's treatment. If no row matches, packets are allowed by default. (Default-deny treatment is possible: add a rule with **priority** 0, **1** as **match**, and **drop** as **action**.)

Summary:

priority	integer, in range 0 to 32,767
match	string
action	string, one of **allow**, **drop**, or **reroute**
nexthop	optional string
nexthops	set of strings
options : pkt_mark	optional string
Common Columns:	
external_ids	map of string-string pairs

Details:

priority: integer, in range 0 to 32,767

The routing policy's priority. Rules with numerically higher priority take precedence over those with lower. A rule is uniquely identified by the priority and match string.

match: string

The packets that the routing policy should match, in the same expression language used for the **match** column in the OVN Southbound database's **Logical_Flow** table.

By default all traffic is allowed. When writing a more restrictive policy, it is important to remember to allow flows such as ARP and IPv6 neighbor discovery packets.

action: string, one of **allow**, **drop**, or **reroute**

The action to take when the routing policy matches:

- **allow**: Forward the packet.
- **drop**: Silently drop the packet.
- **reroute**: Reroute packet to **nexthop** or **nexthops**.

nexthop: optional string

Note: This column is deprecated in favor of **nexthops**.

Next-hop IP address for this route, which should be the IP address of a connected router port or the IP address of a logical port.

nexthops: set of strings

Next-hop ECMP IP addresses for this route. Each IP in the list should be the IP address of a connected router port or the IP address of a logical port.

One IP from the list is selected as next hop.

options : pkt_mark: optional string

Marks the packet with the value specified when the router policy is applied. CMS can inspect this packet marker and take some decisions if desired. This value is not preserved when the packet goes out on the wire.

Common Columns:

external_ids: map of string-string pairs

See **External IDs** at the beginning of this document.

NAT TABLE

Each record represents a NAT rule.

Summary:

type	string, one of **dnat**, **dnat_and_snat**, or **snat**
external_ip	string
external_mac	optional string
external_port_range	string
logical_ip	string
logical_port	optional string
allowed_ext_ips	optional **Address_Set**
exempted_ext_ips	optional **Address_Set**
options : stateless	optional string
Common Columns:	
external_ids	map of string-string pairs

Details:

type: string, one of **dnat**, **dnat_and_snat**, or **snat**

Type of the NAT rule.

- When **type** is **dnat**, the externally visible IP address **external_ip** is DNATted to the IP address **logical_ip** in the logical space.
- When **type** is **snat**, IP packets with their source IP address that either matches the IP address in **logical_ip** or is in the network provided by **logical_ip** is SNATed into the IP address in **external_ip**.
- When **type** is **dnat_and_snat**, the externally visible IP address **external_ip** is DNATted to the IP address **logical_ip** in the logical space. In addition, IP packets with the source IP address that matches **logical_ip** is SNATed into the IP address in **external_ip**.

external_ip: string

An IPv4 address.

external_mac: optional string

A MAC address.

This is only used on the gateway port on distributed routers. This must be specified in order for the NAT rule to be processed in a distributed manner on all chassis. If this is not specified for a NAT rule on a distributed router, then this NAT rule will be processed in a centralized manner on the gateway port instance on the gateway chassis.

This MAC address must be unique on the logical switch that the gateway port is attached to. If the MAC address used on the **logical_port** is globally unique, then that MAC address can be specified as this **external_mac**.

external_port_range: string

L4 source port range

Range of ports, from which a port number will be picked that will replace the source port of to be NATed packet. This is basically PAT (port address translation).

Value of the column is in the format, port_lo-port_hi. For example: external_port_range : "1−30000"

Valid range of ports is 1−65535.

logical_ip: string

An IPv4 network (e.g 192.168.1.0/24) or an IPv4 address.

logical_port: optional string

The name of the logical port where the **logical_ip** resides.

This is only used on distributed routers. This must be specified in order for the NAT rule to be processed in a distributed manner on all chassis. If this is not specified for a NAT rule on a distributed router, then this NAT rule will be processed in a centralized manner on the gateway port instance on the gateway chassis.

allowed_ext_ips: optional **Address_Set**

It represents Address Set of external ips that NAT rule is applicable to. For SNAT type NAT rules, this refers to destination addresses. For DNAT type NAT rules, this refers to source addresses.

This configuration overrides the default NAT behavior of applying a rule solely based on internal IP. Without this configuration, NAT happens without considering the external IP (i.e dest/source for snat/dnat type rule). With this configuration NAT rule is applied ONLY if external ip is in the input Address Set.

exempted_ext_ips: optional **Address_Set**

It represents Address Set of external ips that NAT rule is NOT applicable to. For SNAT type NAT rules, this refers to destination addresses. For DNAT type NAT rules, this refers to source addresses.

This configuration overrides the default NAT behavior of applying a rule solely based on internal IP. Without this configuration, NAT happens without considering the external IP (i.e dest/source for snat/dnat type rule). With this configuration NAT rule is NOT applied if external ip is in the input Address Set.

If there are NAT rules in a logical router with overlapping IP prefixes (including /32), then usage of *exempted_ext_ips* should be avoided in following scenario. a. SNAT rule (let us say RULE1) with logical_ip PREFIX/MASK (let us say 50.0.0.0/24). b. SNAT rule (let us say RULE2) with logical_ip PREFIX/MASK+1 (let us say 50.0.0.0/25). c. Now, if exempted_ext_ips is associated with RULE2, then a logical ip which matches both 50.0.0.0/24 and 50.0.0.0/25 may get the RULE2 applied to it instead of RULE1.

allowed_ext_ips and *exempted_ext_ips* are mutually exclusive to each other. If both Address Sets are set for a rule, then the NAT rule is not considered.

options : stateless: optional string

Indicates if a dnat_and_snat rule should lead to connection tracking state or not.

Common Columns:

external_ids: map of string-string pairs

See **External IDs** at the beginning of this document.

DHCP_Options TABLE

OVN implements native DHCPv4 support which caters to the common use case of providing an IPv4 address to a booting instance by providing stateless replies to DHCPv4 requests based on statically configured address mappings. To do this it allows a short list of DHCPv4 options to be configured and applied at each compute host running **ovn−controller**.

OVN also implements native DHCPv6 support which provides stateless replies to DHCPv6 requests.

Summary:

cidr	string
DHCPv4 options:	
Mandatory DHCPv4 options:	
options : server_id	optional string
options : server_mac	optional string
options : lease_time	optional string, containing an integer, in range 0 to 4,294,967,295
IPv4 DHCP Options:	
options : router	optional string
options : netmask	optional string
options : dns_server	optional string
options : log_server	optional string
options : lpr_server	optional string
options : swap_server	optional string
options : policy_filter	optional string
options : router_solicitation	optional string
options : nis_server	optional string
options : ntp_server	optional string
options : netbios_name_server	optional string
options : classless_static_route	optional string
options : ms_classless_static_route	optional string
Boolean DHCP Options:	
options : ip_forward_enable	optional string, either **0** or **1**
options : router_discovery	optional string, either **0** or **1**
options : ethernet_encap	optional string, either **0** or **1**
Integer DHCP Options:	
options : default_ttl	optional string, containing an integer, in range 0 to 255
options : tcp_ttl	optional string, containing an integer, in range 0 to 255
options : mtu	optional string, containing an integer, in range 68 to 65,535
options : T1	optional string, containing an integer, in range 68 to 4,294,967,295
options : T2	optional string, containing an integer, in range 68 to 4,294,967,295
options : arp_cache_timeout	optional string, containing an integer, in range 0 to 255
options : tcp_keepalive_interval	optional string, containing an integer, in range 0 to 255
options : netbios_node_type	optional string, containing an integer, in range 0 to 255
String DHCP Options:	
options : wpad	optional string
options : bootfile_name	optional string
options : path_prefix	optional string

options : tftp_server_address	optional string
options : domain_name	optional string
options : bootfile_name_alt	optional string
options : broadcast_address	optional string
DHCP Options of type host_id:	
options : tftp_server	optional string
DHCP Options of type domains:	
options : domain_search_list	optional string
DHCPv6 options:	
Mandatory DHCPv6 options:	
options : server_id	optional string
IPv6 DHCPv6 options:	
options : dns_server	optional string
String DHCPv6 options:	
options : domain_search	optional string
options : dhcpv6_stateless	optional string
Common Columns:	
external_ids	map of string-string pairs

Details:

cidr: string
The DHCPv4/DHCPv6 options will be included if the logical port has its IP address in this **cidr**.

DHCPv4 options:

The CMS should define the set of DHCPv4 options as key/value pairs in the **options** column of this table. For **ovn−controller** to include these DHCPv4 options, the **dhcpv4_options** of **Logical_Switch_Port** should refer to an entry in this table.

Mandatory DHCPv4 options:

The following options must be defined.

options : server_id: optional string
The IP address for the DHCP server to use. This should be in the subnet of the offered IP. This is also included in the DHCP offer as option 54, ‘‘server identifier.’’

options : server_mac: optional string
The Ethernet address for the DHCP server to use.

options : lease_time: optional string, containing an integer, in range 0 to 4,294,967,295
The offered lease time in seconds,

The DHCPv4 option code for this option is 51.

IPv4 DHCP Options:

Below are the supported DHCPv4 options whose values are an IPv4 address, e.g. **192.168.1.1**. Some options accept multiple IPv4 addresses enclosed within curly braces, e.g. **{192.168.1.2, 192.168.1.3}**. Please refer to RFC 2132 for more details on DHCPv4 options and their codes.

options : router: optional string
The IP address of a gateway for the client to use. This should be in the subnet of the offered IP. The DHCPv4 option code for this option is 3.

options : netmask: optional string
The DHCPv4 option code for this option is 1.

options : dns_server: optional string
The DHCPv4 option code for this option is 6.

options : log_server: optional string
The DHCPv4 option code for this option is 7.

options : lpr_server: optional string
The DHCPv4 option code for this option is 9.

options : swap_server: optional string
The DHCPv4 option code for this option is 16.

options : policy_filter: optional string
The DHCPv4 option code for this option is 21.

options : router_solicitation: optional string
The DHCPv4 option code for this option is 32.

options : nis_server: optional string
The DHCPv4 option code for this option is 41.

options : ntp_server: optional string
The DHCPv4 option code for this option is 42.

options : netbios_name_server: optional string
The DHCPv4 option code for this option is 44.

options : classless_static_route: optional string
The DHCPv4 option code for this option is 121.

This option can contain one or more static routes, each of which consists of a destination descriptor and the IP address of the router that should be used to reach that destination. Please see RFC 3442 for more details.

Example: **{30.0.0.0/24,10.0.0.10, 0.0.0.0/0,10.0.0.1}**

options : ms_classless_static_route: optional string
The DHCPv4 option code for this option is 249. This option is similar to **classless_static_route** supported by Microsoft Windows DHCPv4 clients.

Boolean DHCP Options:

These options accept a Boolean value, expressed as **0** for false or **1** for true.

options : ip_forward_enable: optional string, either **0** or **1**
The DHCPv4 option code for this option is 19.

options : router_discovery: optional string, either **0** or **1**
The DHCPv4 option code for this option is 31.

options : ethernet_encap: optional string, either **0** or **1**
The DHCPv4 option code for this option is 36.

Integer DHCP Options:

These options accept a nonnegative integer value.

options : default_ttl: optional string, containing an integer, in range 0 to 255
The DHCPv4 option code for this option is 23.

options : tcp_ttl: optional string, containing an integer, in range 0 to 255
The DHCPv4 option code for this option is 37.

options : mtu: optional string, containing an integer, in range 68 to 65,535
The DHCPv4 option code for this option is 26.

options : T1: optional string, containing an integer, in range 68 to 4,294,967,295
This specifies the time interval from address assignment until the client begins trying to renew its address. The DHCPv4 option code for this option is 58.

options : T2: optional string, containing an integer, in range 68 to 4,294,967,295
This specifies the time interval from address assignment until the client begins trying to rebind its address. The DHCPv4 option code for this option is 59.

options : arp_cache_timeout: optional string, containing an integer, in range 0 to 255
The DHCPv4 option code for this option is 35. This option specifies the timeout in seconds for ARP cache entries.

options : tcp_keepalive_interval: optional string, containing an integer, in range 0 to 255
The DHCPv4 option code for this option is 38. This option specifies the interval that the client TCP should wait before sending a keepalive message on a TCP connection.

options : netbios_node_type: optional string, containing an integer, in range 0 to 255
The DHCPv4 option code for this option is 46.

String DHCP Options:

These options accept a string value.

options : wpad: optional string
The DHCPv4 option code for this option is 252. This option is used as part of web proxy auto discovery to provide a URL for a web proxy.

options : bootfile_name: optional string
The DHCPv4 option code for this option is 67. This option is used to identify a bootfile.

options : path_prefix: optional string
The DHCPv4 option code for this option is 210. In PXELINUX' case this option is used to set a common path prefix, instead of deriving it from the bootfile name.

options : tftp_server_address: optional string
The DHCPv4 option code for this option is 150. The option contains one or more IPv4 addresses that the client MAY use. This option is Cisco proprietary, the IEEE standard that matches with this requirement is option 66 (tftp_server).

options : domain_name: optional string
The DHCPv4 option code for this option is 15. This option specifies the domain name that client should use when resolving hostnames via the Domain Name System.

options : bootfile_name_alt: optional string
"bootfile_name_alt" option is used to support iPXE. When both "bootfile_name" and "bootfile_name_alt" are provided by the CMS, "bootfile_name" will be used for option 67 if the dhcp request contains etherboot option (175), otherwise "bootfile_name_alt" will be used.

options : broadcast_address: optional string
The DHCPv4 option code for this option is 28. This option specifies the IP address used as a broadcast address.

DHCP Options of type host_id:

These options accept either an IPv4 address or a string value.

options : tftp_server: optional string
The DHCPv4 option code for this option is 66.

DHCP Options of type domains:

These options accept string value which is a comma separated list of domain names. The domain names are encoded based on RFC 1035.

options : domain_search_list: optional string
The DHCPv4 option code for this option is 119.

DHCPv6 options:

OVN also implements native DHCPv6 support. The CMS should define the set of DHCPv6 options as

key/value pairs. The define DHCPv6 options will be included in the DHCPv6 response to the DHCPv6 Solicit/Request/Confirm packet from the logical ports having the IPv6 addresses in the **cidr**.

Mandatory DHCPv6 options:

The following options must be defined.

options : server_id: optional string
The Ethernet address for the DHCP server to use. This is also included in the DHCPv6 reply as option 2, ‘‘Server Identifier’’ to carry a DUID identifying a server between a client and a server. **ovn−controller** defines DUID based on Link-layer Address [DUID-LL].

IPv6 DHCPv6 options:

Below are the supported DHCPv6 options whose values are an IPv6 address, e.g. **aef0::4**. Some options accept multiple IPv6 addresses enclosed within curly braces, e.g. **{aef0::4, aef0::5}**. Please refer to RFC 3315 for more details on DHCPv6 options and their codes.

options : dns_server: optional string
The DHCPv6 option code for this option is 23. This option specifies the DNS servers that the VM should use.

String DHCPv6 options:

These options accept string values.

options : domain_search: optional string
The DHCPv6 option code for this option is 24. This option specifies the domain search list the client should use to resolve hostnames with DNS.

Example: **"ovn.org"**.

options : dhcpv6_stateless: optional string
This option specifies the OVN native DHCPv6 will work in stateless mode, which means OVN native DHCPv6 will not offer IPv6 addresses for VM/VIF ports, but only reply other configurations, such as DNS and domain search list. When setting this option with string value "true", VM/VIF will configure IPv6 addresses by stateless way. Default value for this option is false.

Common Columns:

external_ids: map of string-string pairs
See **External IDs** at the beginning of this document.

Connection TABLE

Configuration for a database connection to an Open vSwitch database (OVSDB) client.

This table primarily configures the Open vSwitch database server (**ovsdb−server**).

The Open vSwitch database server can initiate and maintain active connections to remote clients. It can also listen for database connections.

Summary:

Core Features:	
target	string (must be unique within table)
Client Failure Detection and Handling:	
max_backoff	optional integer, at least 1,000
inactivity_probe	optional integer
Status:	
is_connected	boolean
status : last_error	optional string
status : state	optional string, one of **ACTIVE**, **BACKOFF**, **CONNECTING**, **IDLE**, or **VOID**
status : sec_since_connect	optional string, containing an integer, at least 0
status : sec_since_disconnect	optional string, containing an integer, at least 0
status : locks_held	optional string
status : locks_waiting	optional string
status : locks_lost	optional string
status : n_connections	optional string, containing an integer, at least 2
status : bound_port	optional string, containing an integer
Common Columns:	
external_ids	map of string-string pairs
other_config	map of string-string pairs

Details:

Core Features:

target: string (must be unique within table)

Connection methods for clients.

The following connection methods are currently supported:

ssl:*host*[**:***port*]

The specified SSL *port* on the host at the given *host*, which can either be a DNS name (if built with unbound library) or an IP address. A valid SSL configuration must be provided when this form is used, this configuration can be specified via command-line options or the **SSL** table.

If *port* is not specified, it defaults to 6640.

SSL support is an optional feature that is not always built as part of Open vSwitch.

tcp:*host*[**:***port*]

The specified TCP *port* on the host at the given *host*, which can either be a DNS name (if built with unbound library) or an IP address. If *host* is an IPv6 address, wrap it in square brackets, e.g. **tcp:[::1]:6640**.

If *port* is not specified, it defaults to 6640.

pssl:[*port*][**:***host*]

Listens for SSL connections on the specified TCP *port*. Specify 0 for *port* to have the kernel automatically choose an available port. If *host*, which can either be a DNS name (if built with unbound library) or an IP address, is specified, then connections are restricted to the resolved or specified local IPaddress (either IPv4 or IPv6 address). If *host* is an IPv6 address, wrap in square brackets, e.g. **pssl:6640:[::1]**. If *host* is not specified then it listens only on IPv4 (but not IPv6) addresses. A valid SSL configuration must be

provided when this form is used, this can be specified either via command-line options or the **SSL** table.

If *port* is not specified, it defaults to 6640.

SSL support is an optional feature that is not always built as part of Open vSwitch.

ptcp:[*port*][**:***host*]
Listens for connections on the specified TCP *port*. Specify 0 for *port* to have the kernel automatically choose an available port. If *host*, which can either be a DNS name (if built with unbound library) or an IP address, is specified, then connections are restricted to the resolved or specified local IP address (either IPv4 or IPv6 address). If *host* is an IPv6 address, wrap it in square brackets, e.g. **ptcp:6640:[::1]**. If *host* is not specified then it listens only on IPv4 addresses.

If *port* is not specified, it defaults to 6640.

When multiple clients are configured, the **target** values must be unique. Duplicate **target** values yield unspecified results.

Client Failure Detection and Handling:

max_backoff: optional integer, at least 1,000
Maximum number of milliseconds to wait between connection attempts. Default is implementation-specific.

inactivity_probe: optional integer
Maximum number of milliseconds of idle time on connection to the client before sending an inactivity probe message. If Open vSwitch does not communicate with the client for the specified number of seconds, it will send a probe. If a response is not received for the same additional amount of time, Open vSwitch assumes the connection has been broken and attempts to reconnect. Default is implementation-specific. A value of 0 disables inactivity probes.

Status:

Key-value pair of **is_connected** is always updated. Other key-value pairs in the status columns may be updated depends on the **target** type.

When **target** specifies a connection method that listens for inbound connections (e.g. **ptcp:** or **punix:**), both **n_connections** and **is_connected** may also be updated while the remaining key-value pairs are omitted.

On the other hand, when **target** specifies an outbound connection, all key-value pairs may be updated, except the above-mentioned two key-value pairs associated with inbound connection targets. They are omitted.

is_connected: boolean
true if currently connected to this client, **false** otherwise.

status : last_error: optional string
A human-readable description of the last error on the connection to the manager; i.e. **strerror(errno)**. This key will exist only if an error has occurred.

status : state: optional string, one of **ACTIVE**, **BACKOFF**, **CONNECTING**, **IDLE**, or **VOID**
The state of the connection to the manager:

VOID Connection is disabled.

BACKOFF
Attempting to reconnect at an increasing period.

CONNECTING
Attempting to connect.

ACTIVE
Connected, remote host responsive.

IDLE Connection is idle. Waiting for response to keep-alive.

These values may change in the future. They are provided only for human consumption.

status : sec_since_connect: optional string, containing an integer, at least 0
The amount of time since this client last successfully connected to the database (in seconds). Value is empty if client has never successfully been connected.

status : sec_since_disconnect: optional string, containing an integer, at least 0
The amount of time since this client last disconnected from the database (in seconds). Value is empty if client has never disconnected.

status : locks_held: optional string
Space-separated list of the names of OVSDB locks that the connection holds. Omitted if the connection does not hold any locks.

status : locks_waiting: optional string
Space-separated list of the names of OVSDB locks that the connection is currently waiting to acquire. Omitted if the connection is not waiting for any locks.

status : locks_lost: optional string
Space-separated list of the names of OVSDB locks that the connection has had stolen by another OVSDB client. Omitted if no locks have been stolen from this connection.

status : n_connections: optional string, containing an integer, at least 2
When **target** specifies a connection method that listens for inbound connections (e.g. **ptcp:** or **pssl:**) and more than one connection is actually active, the value is the number of active connections. Otherwise, this key-value pair is omitted.

status : bound_port: optional string, containing an integer
When **target** is **ptcp:** or **pssl:**, this is the TCP port on which the OVSDB server is listening. (This is particularly useful when **target** specifies a port of 0, allowing the kernel to choose any available port.)

Common Columns:

The overall purpose of these columns is described under **Common Columns** at the beginning of this document.

external_ids: map of string-string pairs

other_config: map of string-string pairs

DNS TABLE

Each row in this table stores the DNS records. The **Logical_Switch** table's **dns_records** references these records.

Summary:

records	map of string-string pairs
external_ids	map of string-string pairs

Details:

records: map of string-string pairs

Key-value pair of DNS records with **DNS query name** as the key and value as a string of IP address(es) separated by comma or space.

Example: "vm1.ovn.org" = "10.0.0.4 aef0::4"

external_ids: map of string-string pairs

See **External IDs** at the beginning of this document.

SSL TABLE

SSL configuration for ovn-nb database access.

Summary:

private_key	string
certificate	string
ca_cert	string
bootstrap_ca_cert	boolean
ssl_protocols	string
ssl_ciphers	string
Common Columns:	
external_ids	map of string-string pairs

Details:

private_key: string

Name of a PEM file containing the private key used as the switch's identity for SSL connections to the controller.

certificate: string

Name of a PEM file containing a certificate, signed by the certificate authority (CA) used by the controller and manager, that certifies the switch's private key, identifying a trustworthy switch.

ca_cert: string

Name of a PEM file containing the CA certificate used to verify that the switch is connected to a trustworthy controller.

bootstrap_ca_cert: boolean

If set to **true**, then Open vSwitch will attempt to obtain the CA certificate from the controller on its first SSL connection and save it to the named PEM file. If it is successful, it will immediately drop the connection and reconnect, and from then on all SSL connections must be authenticated by a certificate signed by the CA certificate thus obtained. **This option exposes the SSL connection to a man−in−the−middle attack obtaining the initial CA certificate.** It may still be useful for bootstrapping.

ssl_protocols: string

List of SSL protocols to be enabled for SSL connections. The default when this option is omitted is **TLSv1,TLSv1.1,TLSv1.2**.

ssl_ciphers: string

List of ciphers (in OpenSSL cipher string format) to be supported for SSL connections. The default when this option is omitted is **HIGH:!aNULL:!MD5**.

Common Columns:

The overall purpose of these columns is described under **Common Columns** at the beginning of this document.

external_ids: map of string-string pairs

Gateway_Chassis TABLE

Association of a chassis to a logical router port. The traffic going out through an specific router port will be redirected to a chassis, or a set of them in high availability configurations.

Summary:

name	string (must be unique within table)
chassis_name	string
priority	integer, in range 0 to 32,767
options	map of string-string pairs
Common Columns:	
external_ids	map of string-string pairs

Details:

name: string (must be unique within table)

Name of the **Gateway_Chassis**.

A suggested, but not required naming convention is **${port_name}_${chassis_name}**.

chassis_name: string

Name of the chassis that we want to redirect traffic through for the associated logical router port. The value must match the **name** column of the **Chassis** table in the **OVN_Southbound** database.

priority: integer, in range 0 to 32,767

This is the priority of a chassis among all **Gateway_Chassis** belonging to the same logical router port.

options: map of string-string pairs

Reserved for future use.

Common Columns:

external_ids: map of string-string pairs

See **External IDs** at the beginning of this document.

HA_Chassis_Group TABLE

Table representing a group of chassis which can provide high availability services. Each chassis in the group is represented by the table **HA_Chassis**. The HA chassis with highest priority will be the master of this group. If the master chassis failover is detected, the HA chassis with the next higher priority takes over the responsibility of providing the HA. If a distributed gateway router port references a row in this table, then the master HA chassis in this group provides the gateway functionality.

Summary:

name	string (must be unique within table)
ha_chassis	set of **HA_Chassis**es
Common Columns:	
external_ids	map of string-string pairs

Details:

name: string (must be unique within table)

Name of the **HA_Chassis_Group**. Name should be unique.

ha_chassis: set of **HA_Chassis**es

A list of HA chassis which belongs to this group.

Common Columns:

external_ids: map of string-string pairs

See **External IDs** at the beginning of this document.

HA_Chassis TABLE

Summary:

chassis_name	string
priority	integer, in range 0 to 32,767
Common Columns:	
external_ids	map of string-string pairs

Details:

chassis_name: string
: Name of the chassis which is part of the HA chassis group. The value must match the **name** column of the **Chassis** table in the **OVN_Southbound** database.

priority: integer, in range 0 to 32,767
: Priority of the chassis. Chassis with highest priority will be the master.

Common Columns:

external_ids: map of string-string pairs
: See **External IDs** at the beginning of this document.

BFD TABLE

Contains BFD parameter for ovn-controller BFD configuration. OVN BFD implementation is used to provide detection of failures in the path between adjacent forwarding engines, including the OVN interfaces. OVN BFD provides link status info to OVN northd in order to update logical flows according to the status of BFD endpoints. In the current implementation OVN BFD is used to check next-hop status for ECMP routes. Please note BFD table refers to OVN BFD implementation and not to OVS legacy one.

Summary:

Configuration:

logical_port	string
dst_ip	string
min_tx	optional integer, at least 1
min_rx	optional integer
detect_mult	optional integer, at least 1
options	map of string-string pairs
external_ids	map of string-string pairs

Status Reporting:

status	optional string, one of **admin_down**, **down**, **init**, or **up**

Details:

Configuration:

ovn−northd reads configuration from these columns.

logical_port: string
: OVN logical port when BFD engine is running.

dst_ip: string
: BFD peer IP address.

min_tx: optional integer, at least 1
: This is the minimum interval, in milliseconds, that the local system would like to use when transmitting BFD Control packets, less any jitter applied. The value zero is reserved. Default value is 1000 ms.

min_rx: optional integer
: This is the minimum interval, in milliseconds, between received BFD Control packets that this system is capable of supporting, less any jitter applied by the sender. If this value is zero, the transmitting system does not want the remote system to send any periodic BFD Control packets.

detect_mult: optional integer, at least 1
: Detection time multiplier. The negotiated transmit interval, multiplied by this value, provides the Detection Time for the receiving system in Asynchronous mode. Default value is 5.

options: map of string-string pairs
: Reserved for future use.

external_ids: map of string-string pairs
: See **External IDs** at the beginning of this document.

Status Reporting:

ovn−northd writes BFD status into these columns.

status: optional string, one of **admin_down**, **down**, **init**, or **up**
: BFD port logical states. Possible values are:

- **admin_down**
- **down**
- **init**

- **up**