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.

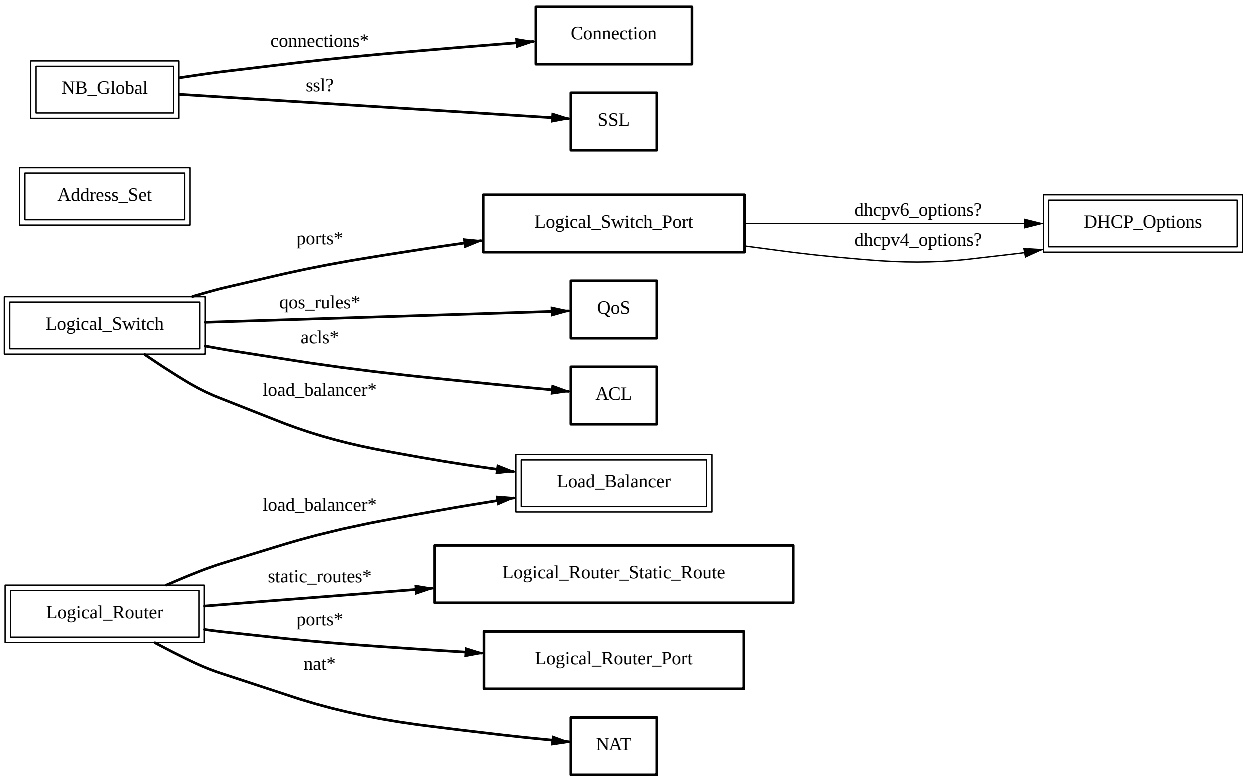
TABLE SUMMARY

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
Address_Set	Address Sets
Load_Balancer	load balancer
ACL	Access Control List (ACL) rule
Logical_Router	L3 logical router
QoS	QOS table
Logical_Router_Port	L3 logical router port
Logical_Router_Static_Route	Logical router static routes
NAT	NAT rules
DHCP_Options	DHCP options
Connection	OVSDB client connections.
SSL	SSL configuration.

TABLE RELATIONSHIPS

The following diagram shows the relationship among tables in the database. Each node represents a table. Tables that are part of the ‘‘root set’’ are shown with double borders. Each edge leads from the table that contains it and points to the table that its value represents. Edges are labeled with their column names, followed by a constraint on the number of allowed values: **?** for zero or one, ***** for zero or more, **+** for one or more. Thick lines represent strong references; thin lines represent weak references.



NB_Global TABLE

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

Summary:

Status:	
nb_cfg	integer
sb_cfg	integer
hv_cfg	integer
Common Columns:	
external_ids	map of string-string pairs
Connection Options:	
connections	set of **Connection**s
ssl	optional **SSL**

Details:

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.

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.

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** 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.

Common Columns:

external_ids: map of string-string pairs

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

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.

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 a physical network (bridged logical switches). They work in the same way when providing connectivity between logical ports on same chasis, 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 segment with the help of **localnet** ports. Each bridged logical switch has one and only one **localnet** port, which has only one special address **unknown**.

Summary:

name	string
ports	set of **Logical_Switch_Port**s
load_balancer	set of **Load_Balancer**s
acls	set of **ACL**s
qos_rules	set of **QoS**s
other_config:	
other_config : subnet	optional string
Common Columns:	
external_ids	map of string-string pairs

Details:

name: string

A name for the logical switch. This name has no special meaning or purpose other than to provide convenience for human interaction with the ovn-nb database. There is no requirement for the name to be unique. The logical switch’s UUID should be used as the unique identifier.

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 **Load_Balancer**s

Load balance a virtual ipv4 address to a set of logical port endpoint ipv4 addresses.

acls: set of **ACL**s

Access control rules that apply to packets within the logical switch.

qos_rules: set of **QoS**s

QOS marking rules that apply to packets within the logical switch.

other_config:

Additional configuration options for the logical switch.

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. Use the **dynamic** keyword in the **Logical_Switch_Port** table’s **addresses** column to request dynamic address assignment for a particular port.

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 : qos_max_rate	optional string
options : qos_burst	optional string
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
Common Columns:	
dhcpv4_options	optional weak reference to **DHCP_Options**
dhcpv6_options	optional weak reference to **DHCP_Options**
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.

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.

localnet
A connection to a locally accessible network from each **ovn−controller** instance. A logical switch can only have a single **localnet** port attached. This is used to model direct connectivity to an existing network.

l2gateway
A connection to a physical network.

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

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
MAC address of the **router−port** followed by a list of SNAT and DNAT IP addresses. This is used to send gratuitous ARPs for SNAT and DNAT IP addresses via **localnet** and is valid for only L3 gateway ports. 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.

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.

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 : 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.

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**. 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.

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 this keyword to make **ovn−northd** generate a globally unique MAC address and choose an unused IPv4 address with the logical port's subnet and store them in the port's **dynamic_addresses** column. **ovn−northd** will use the subnet specified in **other_config:subnet** in the port's **Logical_Switch**.

Ethernet address followed by keyword "dynamic"
The keyword **dynamic** after the MAC address indicates that **ovn−northd** should choose an unused IPv4 address from the logical port's subnet and store it with the specified MAC in the port's **dynamic_addresses** column. **ovn−northd** will use the subnet specified in **other_config:subnet** in the port's **Logical_Switch** table.

Examples:

80:fa:5b:06:72:b7 dynamic
This indicates that the logical port owns the specified MAC address and **ovn−northd** should allocate an unused IPv4 address for the logical port from the corresponding logical switch subnet.

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 **redirect−chassis** 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.

Common Columns:

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.

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.

Load_Balancer TABLE

Each row represents one load balancer.

Summary:

name	string
vips	map of string-string pairs
protocol	optional string, either **udp** or **tcp**
Common Columns:	
external_ids	map of string-string pairs

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 IPv4 addresses (and an optional port number with **:** as a separator) associated with this load balancer and their corresponding endpoint IPv4 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 IPv4 address (and port number) provided here as a key, then OVN will statefully replace the destination IP address by one of the provided IPv4 address (and port number) in this map as a value. Examples for keys are "192.168.1.4" and "172.16.1.8:80". Examples for value are "10.0.0.1, 10.0.0.2" and "20.0.0.10:8800, 20.0.0.11:8800".

protocol: optional string, either **udp** or **tcp**

Valid protocols are **tcp** or **udp**. 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**.

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 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, **0** as **match**, and **deny** as **action**.)

Summary:

priority	integer, in range 0 to 32,767
direction	string, either **to−lport** or **from−lport**
match	string
action	string, one of **allow−related**, **drop**, **allow**, or **reject**
log	boolean
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 **to−lport** or **from−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.

Note that when **localnet** port exists in a lswitch, for **to−lport** direction, the **inport** works only if the **to−lport** is located on the same chassis as the **inport**.

action: string, one of **allow−related**, **drop**, **allow**, 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 ICMP unreachable message for other IP-based protocols. **Not implemented−−currently treated as drop**

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**.

Logging is not yet implemented.

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:

name	string
ports	set of **Logical_Router_Port**s
static_routes	set of **Logical_Router_Static_Route**s
enabled	optional boolean
nat	set of **NAT**s
load_balancer	set of **Load_Balancer**s
Options:	
options : chassis	optional string
options : dnat_force_snat_ip	optional string
options : lb_force_snat_ip	optional string
Common Columns:	
external_ids	map of string-string pairs

Details:

name: string

A name for the logical router. This name has no special meaning or purpose other than to provide convenience for human interaction with the ovn-nb database. There is no requirement for the name to be unique. The logical router's UUID should be used as the unique identifier.

ports: set of **Logical_Router_Port**s

The router's ports.

static_routes: set of **Logical_Router_Static_Route**s

One or more static routes 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 one logical router port with a **redirect−chassis** specified.

load_balancer: set of **Load_Balancer**s

Load balance a virtual ipv4 address to a set of logical port ipv4 addresses. Load balancer rules only work on the Gateway routers.

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 the IP address 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 IP address.

options : lb_force_snat_ip: optional string

If set, indicates the IP address 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 router, 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 IP address.

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. The **action** column for the highest-**priority** matching row in this table determines a packet's qos marking. If no row matches, packets will not have any qos marking.

Summary:

priority	integer, in range 0 to 32,767
direction	string, either **to−lport** or **from−lport**
match	string
action	map of 1 to 1 string-integer pairs, key must be **dscp**, value in range 0 to 63
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 **to−lport** or **from−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 1 to 1 string-integer pairs, key must be **dscp**, value in range 0 to 63

The action to be performed on the matched packet

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

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
Options:	
options : redirect-chassis	optional string
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 ovn-nb database, this column is used as reference by its patch port in **Logical_Switch_Port** or another logical router port in **Logical_Router_Port**.

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.

Options:

Additional options for the logical router port.

options : redirect-chassis: optional string

If set, this indicates that this logical router port represents a distributed gateway port that connects this router to a logical switch with a localnet port. There may be at most one such logical router port on each logical router.

Even when a **redirect−chassis** is specified, 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 specified **redirect−chassis**.

When this option is specified, the peer logical switch port's **addresses** must be set to **router**. With this setting, the **external_mac**s specified in NAT rules are automatically programmed in 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 **redirect−chassis**.

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.

Summary:

ip_prefix	string
policy	optional string, either **src−ip** or **dst−ip**
nexthop	string
output_port	optional string

Details:

ip_prefix: string

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

policy: optional string, either **src−ip** or **dst−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**.

NAT TABLE

Each record represents a NAT rule.

Summary:

type	string, one of **snat**, **dnat**, or **dnat_and_snat**
external_ip	string
external_mac	optional string
logical_ip	string
logical_port	optional string

Details:

type: string, one of **snat**, **dnat**, or **dnat_and_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 **redirect−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**.

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 **redirect−chassis**.

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 : router	optional string
options : lease_time	optional string, containing an integer, in range 0 to 4,294,967,295
IPv4 DHCP Options:	
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 : tftp_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 **1** or **0**
options : router_discovery	optional string, either **1** or **0**
options : ethernet_encap	optional string, either **1** or **0**
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
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 : 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 : 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 : 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 : tftp_server: optional string

The DHCPv4 option code for this option is 66.

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 **1** or **0**
The DHCPv4 option code for this option is 19.

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

options : ethernet_encap: optional string, either **1** or **0**
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.

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**, **VOID**, **CONNECTING**, **IDLE**, or **BACKOFF**
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:*ip*[**:***port*]

The specified SSL *port* on the host at the given *ip*, which must be expressed as an IP address (not a DNS name). 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:*ip*[**:***port*]

The specified TCP *port* on the host at the given *ip*, which must be expressed as an IP address (not a DNS name), where *ip* can be IPv4 or IPv6 address. If *ip* 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*][**:***ip*]

Listens for SSL connections on the specified TCP *port*. Specify 0 for *port* to have the kernel automatically choose an available port. If *ip*, which must be expressed as an IP address (not a DNS name), is specified, then connections are restricted to the specified local IP address (either IPv4 or IPv6 address). If *ip* is an IPv6 address, wrap in square brackets, e.g. **pssl:6640:[::1]**. If *ip* 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*][**:***ip*]
Listens for connections on the specified TCP *port*. Specify 0 for *port* to have the kernel automatically choose an available port. If *ip*, which must be expressed as an IP address (not a DNS name), is specified, then connections are restricted to the specified local IP address (either IPv4 or IPv6 address). If *ip* is an IPv6 address, wrap it in square brackets, e.g. **ptcp:6640:[::1]**. If *ip* 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**, **VOID**, **CONNECTING**, **IDLE**, or **BACKOFF**
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

SSL TABLE

SSL configuration for ovn-nb database access.

Summary:

private_key	string
certificate	string
ca_cert	string
bootstrap_ca_cert	boolean
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.

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