NAME

ovn-nbctl − Open Virtual Network northbound db management utility

SYNOPSIS

ovn−nbctl [*options*] *command* [*arg...*]

DESCRIPTION

This utility can be used to manage the OVN northbound database.

GENERAL COMMANDS

init Initializes the database, if it is empty. If the database has already been initialized, this command has no effect.

show [*switch* **|** *router***]**
Prints a brief overview of the database contents. If *switch* is provided, only records related to that logical switch are shown. If *router* is provided, only records related to that logical router are shown.

LOGICAL SWITCH COMMANDS

ls−add Creates a new, unnamed logical switch, which initially has no ports. The switch does not have a name, other commands must refer to this switch by its UUID.

[**−−may−exist** | **−−add−duplicate**] **ls−add** *switch*
Creates a new logical switch named *switch*, which initially has no ports.

The OVN northbound database schema does not require logical switch names to be unique, but the whole point to the names is to provide an easy way for humans to refer to the switches, making duplicate names unhelpful. Thus, without any options, this command regards it as an error if *switch* is a duplicate name. With **−−may−exist**, adding a duplicate name succeeds but does not create a new logical switch. With **−−add−duplicate**, the command really creates a new logical switch with a duplicate name. It is an error to specify both options. If there are multiple logical switches with a duplicate name, configure the logical switches using the UUID instead of the *switch* name.

[**−−if−exists**] **ls−del** *switch*
Deletes *switch*. It is an error if *switch* does not exist, unless **−−if−exists** is specified.

ls−list Lists all existing switches on standard output, one per line.

ACL COMMANDS

These commands operates on ACL objects for a given *entity*. The *entity* can be either a logical switch or a port group. The *entity* can be specified as uuid or name. The **−−type** option can be used to specify the type of the *entity*, in case both a logical switch and a port groups exist with the same name specified for *entity*. **type** must be either **switch** or **port−group**.

[**−−type=**{**switch** | **port−group**}] [**−−log**] [**−−meter=***meter*] [**−−severity=***severity*] [**−−name=***name*] [**−−may−exist**] **acl−add** *entity direction priority match verdict*
Adds the specified ACL to *entity*. *direction* must be either **from−lport** or **to−lport**. *priority* must be between **0** and **32767**, inclusive. A full description of the fields are in **ovn−nb**(5). If **−−may−exist** is specified, adding a duplicated ACL succeeds but the ACL is not really created. Without **−−may−exist**, adding a duplicated ACL results in error.

The **−−log** option enables packet logging for the ACL. The options **−−severity** and **−−name** specify a severity and name, respectively, for log entries (and also enable logging). The severity must be one of **alert**, **warning**, **notice**, **info**, or **debug**. If a severity is not specified, the default is **info**. The **−−meter=***meter* option is used to rate-limit packet logging. The *meter* argument names a meter configured by **meter−add**.

[**−−type=**{**switch** | **port−group**}] **acl−del** *entity* [*direction* [*priority match*]]
Deletes ACLs from *entity*. If only *entity* is supplied, all the ACLs from the *entity* are deleted. If *direction* is also specified, then all the flows in that direction will be deleted from the *entity*. If all the fields are given, then a single flow that matches all the fields will

be deleted.

[**−−type=**{**switch** | **port−group**}] **acl−list** *entity*
Lists the ACLs on *entity*.

LOGICAL SWITCH QOS RULE COMMANDS

[**−−may−exist**] **qos−add** *switch direction priority match* [**dscp=***dscp*] [**rate=***rate* [**burst=***burst*]]
Adds QoS marking and metering rules to *switch*. *direction* must be either **from−lport** or **to−lport**. *priority* must be between **0** and **32767**, inclusive.

If **dscp=***dscp* is specified, then matching packets will have DSCP marking applied. *dscp* must be between **0** and **63**, inclusive. If **rate=***rate* is specified then matching packets will have metering applied at *rate* kbps. If metering is configured, then **burst=***burst* specifies the burst rate limit in kilobits. **dscp** and/or **rate** are required arguments.

If **−−may−exist** is specified, adding a duplicated QoS rule succeeds but the QoS rule is not really created. Without **−−may−exist**, adding a duplicated QoS rule results in error.

qos−del *switch* [*direction* [*priority match*]]
Deletes QoS rules from *switch*. If only *switch* is supplied, all the QoS rules from the logical switch are deleted. If *direction* is also specified, then all the flows in that direction will be deleted from the logical switch. If all the fields are supplied, then a single flow that matches the given fields will be deleted.

If *switch* and *uuid* are supplied, then the QoS rule with sepcified uuid is deleted.

qos−list *switch*
Lists the QoS rules on *switch*.

METER COMMANDS

meter−add *name action rate unit* [*burst*]
Adds the specified meter. *name* must be a unique name to identify this meter. The *action* argument specifies what should happen when this meter is exceeded. The only supported action is **drop**.

The *unit* specifies the unit for the *rate* argument; valid values are **kbps** and **pktps** for kilobits per second and packets per second, respectively. The *burst* option configures the maximum burst allowed for the band in kilobits or packets depending on whether the *unit* chosen was **kbps** or **pktps**, respectively. If a burst is not supplied, the switch is free to select some reasonable value depending on its configuration.

ovn−nbctl only supports adding a meter with a single band, but the other commands support meters with multiple bands.

Names that start with "__" (two underscores) are reserved for internal use by OVN, so **ovn−nbctl** does not allow adding them.

meter−del [*name*]
Deletes meters. By default, all meters are deleted. If *name* is supplied, only the meter with that name will be deleted.

meter−list
Lists all meters.

LOGICAL SWITCH PORT COMMANDS

[**−−may−exist**] **lsp−add** *switch port*
Creates on *lswitch* a new logical switch port named *port*.

It is an error if a logical port named *port* already exists, unless **−−may−exist** is specified. Regardless of **−−may−exist**, it is an error if the existing port is in some logical switch other than *switch* or if it has a parent port.

[**−−may−exist**] **lsp−add** *switch port parent tag_request*
Creates on *switch* a logical switch port named *port* that is a child of *parent* that is identified with VLAN ID *tag_request*, which must be between **0** and **4095**, inclusive. If *tag_request* is **0**,

ovn−northd generates a tag that is unique in the scope of *parent*. This is useful in cases such as virtualized container environments where Open vSwitch does not have a direct connection to the container's port and it must be shared with the virtual machine's port.

It is an error if a logical port named *port* already exists, unless **−−may−exist** is specified. Regardless of **−−may−exist**, it is an error if the existing port is not in *switch* or if it does not have the specified *parent* and *tag_request*.

[**−−if−exists**] **lsp−del** *port*
Deletes *port*. It is an error if *port* does not exist, unless **−−if−exists** is specified.

lsp−list *switch*
Lists all the logical switch ports within *switch* on standard output, one per line.

lsp−get−parent *port*
If set, get the parent port of *port*. If not set, print nothing.

lsp−get−tag *port*
If set, get the tag for *port* traffic. If not set, print nothing.

lsp−set−addresses *port* [*address*]...
Sets the addresses associated with *port* to *address*. Each *address* should be one of the following:

an Ethernet address, optionally followed by a space and one or more IP addresses
OVN delivers packets for the Ethernet address to this port.

unknown
OVN delivers unicast Ethernet packets whose destination MAC address is not in any logical port's addresses column to ports with address **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.

router Accepted only when the **type** of the logical switch port 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 **lsp−set−options**.

Multiple addresses may be set. If no *address* argument is given, *port* will have no addresses associated with it.

lsp−get−addresses *port*
Lists all the addresses associated with *port* on standard output, one per line.

lsp−set−port−security *port* [*addrs*]...
Sets the port security addresses associated with *port* to *addrs*. Multiple sets of addresses may be set by using multiple *addrs* arguments. If no *addrs* argument is given, *port* will not have port security enabled.

Port security limits the addresses from which a logical port may send packets and to which it may receive packets. See the **ovn−nb**(5) documentation for the **port_security** column in the **Logical_Switch_Port** table for details.

lsp−get−port−security *port*
Lists all the port security addresses associated with *port* on standard output, one per line.

lsp−get−up *port*
Prints the state of *port*, either **up** or **down**.

lsp−set−enabled *port state*
Set the administrative state of *port*, either **enabled** or **disabled**. When a port is disabled, no traffic is allowed into or out of the port.

lsp−get−enabled *port*
Prints the administrative state of *port*, either **enabled** or **disabled**.

lsp−set−type *port type*
Set the type for the logical port. The type must be one of the following:

(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.

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.

lsp−get−type *port*
Get the type for the logical port.

lsp−set−options *port* [*key*=*value*]...
Set type-specific key-value options for the logical port.

lsp−get−options *port*
Get the type-specific options for the logical port.

lsp−set−dhcpv4−options *port dhcp_options*
Set the DHCPv4 options for the logical port. The *dhcp_options* is a UUID referring to a set of DHCP options in the **DHCP_Options** table.

lsp−get−dhcpv4−options *port*
Get the configured DHCPv4 options for the logical port.

lsp−set−dhcpv6−options *port dhcp_options*
Set the DHCPv6 options for the logical port. The *dhcp_options* is a UUID referring to a set of DHCP options in the **DHCP_Options** table.

lsp−get−dhcpv6−options *port*
Get the configured DHCPv6 options for the logical port.

lsp−get−ls *port*
Get the logical switch which the *port* belongs to.

FORWARDING GROUP COMMANDS

[**−−liveness**]**fwd−group−add** *group switch vip vmac ports*
Creates a new forwarding group named *group* as the name with the provided *vip* and *vmac*. *vip* should be a virtual IP address and *vmac* should be a virtual MAC address to access the forwarding group. *ports* are the logical switch port names that are put in the forwarding group. Example for *ports* is lsp1 lsp2 ... Traffic destined to virtual IP of the forwarding group will be load balanced to all the child ports.

When **−−liveness** is specified then child ports are expected to be bound to external devices like routers. BFD should be configured between hypervisors and the external devices. The child port selection will become dependent on BFD status with its external device.

[**−−if−exists**] **fwd−group−del** *group*
Deletes *group*. It is an error if *group* does not exist, unless **−−if−exists** is specified.

fwd−group−list [*switch*]
Lists all existing forwarding groups, If *switch* is specified then only the forwarding groups configured for *switch* will be listed.

LOGICAL ROUTER COMMANDS

lr−add Creates a new, unnamed logical router, which initially has no ports. The router does not have a name, other commands must refer to this router by its UUID.

[**−−may−exist** | **−−add−duplicate**] **lr−add** *router*
Creates a new logical router named *router*, which initially has no ports.

The OVN northbound database schema does not require logical router names to be unique, but the whole point to the names is to provide an easy way for humans to refer to the routers, making duplicate names unhelpful. Thus, without any options, this command regards it as an error if *router* is a duplicate name. With **−−may−exist**, adding a duplicate name succeeds but does not create a new logical router. With **−−add−duplicate**, the command really creates a new logical router with a duplicate name. It is an error to specify both options. If there are multiple logical routers with a duplicate name, configure the logical routers using the UUID instead of the *router* name.

[**−−if−exists**] **lr−del** *router*
Deletes *router*. It is an error if *router* does not exist, unless **−−if−exists** is specified.

lr−list Lists all existing routers on standard output, one per line.

LOGICAL ROUTER PORT COMMANDS

[**−−may−exist**] **lrp−add** *router port mac network...* [**peer=***peer*]
Creates on *router* a new logical router port named *port* with Ethernet address *mac* and one or more IP address/netmask for each *network*.

The optional argument **peer** identifies a logical router port that connects to this one. The following example adds a router port with an IPv4 and IPv6 address with peer **lr1**:

lrp−add lr0 lrp0 00:11:22:33:44:55 192.168.0.1/24 2001:db8::1/64 peer=lr1

It is an error if a logical router port named *port* already exists, unless **−−may−exist** is specified. Regardless of **−−may−exist**, it is an error if the existing router port is in some logical router other than *router*.

[**−−if−exists**] **lrp−del** *port*
Deletes *port*. It is an error if *port* does not exist, unless **−−if−exists** is specified.

lrp−list *router*
Lists all the logical router ports within *router* on standard output, one per line.

lrp−set−enabled *port state*
Set the administrative state of *port*, either **enabled** or **disabled**. When a port is disabled, no traffic is allowed into or out of the port.

lrp−get−enabled *port*
Prints the administrative state of *port*, either **enabled** or **disabled**.

lrp−set−gateway−chassis *port chassis* [*priority*]
Set gateway chassis for *port*. *chassis* is the name of the chassis. This creates a gateway chassis entry in Gateway_Chassis table. It won't check if chassis really exists in OVN_Southbound database. Priority will be set to 0 if *priority* is not provided by user. *priority* must be between **0** and **32767**, inclusive.

lrp−del−gateway−chassis *port chassis*
Deletes gateway chassis from *port*. It is an error if gateway chassis with *chassis* for *port* does not exist.

lrp−get−gateway−chassis *port*

Lists all the gateway chassis with priority within *port* on standard output, one per line, ordered based on priority.

LOGICAL ROUTER STATIC ROUTE COMMANDS

[**−−may−exist**] [**−−policy**=*POLICY*] [**−−ecmp**] [**−−ecmp−symmetric−reply**] [**−−bfd[=***UUID*]] **lr−route−add** *router prefix nexthop* [*port*]

Adds the specified route to *router*. *prefix* describes an IPv4 or IPv6 prefix for this route, such as **192.168.100.0/24**. *nexthop* specifies the gateway to use for this route, which should be the IP address of one of *router* logical router ports or the IP address of a logical port. If *port* is specified, packets that match this route will be sent out that port. When *port* is omitted, OVN infers the output port based on *nexthop*.

−−policy describes the policy used to make routing decisions. This should be one of "dst-ip" or "src-ip". If not specified, the default is "dst-ip".

The **−−ecmp** option allows for multiple routes with the same *prefix POLICY* but different *nexthop* and *port* to be added.

The **−−ecmp−symmetric−reply** option makes it so that traffic that arrives over an ECMP route will have its reply traffic sent out over that same route. Setting **−−ecmp−symmetric−reply** implies **−−ecmp** so it is not necessary to set both.

−−bfd option is used to link a BFD session to the OVN route. If the BFD session UUID is provided, it will be used for the OVN route otherwise the next-hop will be used to perform a lookup in the OVN BFD table. If the lookup fails and *port* is specified, a new entry in the BFD table will be created using the *nexthop* as *dst_ip* and *port* as *logical_port*.

It is an error if a route with *prefix* and *POLICY* already exists, unless **−−may−exist**, **−−ecmp**, or **−−ecmp−symmetric−reply** is specified. If **−−may−exist** is specified but not **−−ecmp** or **−−ecmp−symmetric−reply**, the existed route will be updated with the new nexthop and port. If **−−ecmp** or **−−ecmp−symmetric−reply** is specified, a new route will be added, regardless of the existed route., which is useful when adding ECMP routes, i.e. routes with same *POLICY* and *prefix* but different *nexthop* and *port*.

[**−−if−exists**] [**−−policy**=*POLICY*] **lr−route−del** *router* [*prefix* [*nexthop* [*port*]]]

Deletes routes from *router*. If only *router* is supplied, all the routes from the logical router are deleted. If *POLICY*, *prefix*, *nexthop* and/or *port* are also specified, then all the routes that match the conditions will be deleted from the logical router.

It is an error if there is no matching route entry, unless **−−if−exists** is specified.

lr−route−list *router*

Lists the routes on *router*.

LOGICAL ROUTER POLICY COMMANDS

[**−−may−exist**]**lr−policy−add** *router priority match action* [*nexthop*[,*nexthop*,...]] [*options key=value]*]

Add Policy to *router* which provides a way to configure permit/deny and reroute policies on the router. Permit/deny policies are similar to OVN ACLs, but exist on the logical-router. Reroute policies are needed for service-insertion and service-chaining. *nexthop* is an optional parameter. It needs to be provided only when *action* is *reroute*. Multiple **nexthops** can be specified for ECMP routing. A policy is uniquely identified by *priority* and *match*. Multiple policies can have the same *priority*. *options* sets the router policy options as key-value pair. The supported option is : **pkt_mark**.

If **−−may−exist** is specified, adding a duplicated routing policy with the same priority and match string is not really created. Without **−−may−exist**, adding a duplicated routing policy results in error.

The following example shows a policy to lr1, which will drop packets from**192.168.100.0/24**.

lr−policy−add lr1 100 ip4.src == 192.168.100.0/24 drop.

lr−policy−add lr1 100 ip4.src == 192.168.100.0/24 allow pkt_mark=100 .

[**−−if−exists**] **lr−policy−del** *router* [*{priority | uuid} [match]*]
Deletes polices from *router*. If only *router* is supplied, all the polices from the logical router are deleted. If *priority* and/or *match* are also specified, then all the polices that match the conditions will be deleted from the logical router.

If *router* and *uuid* are supplied, then the policy with sepcified uuid is deleted. It is an error if *uuid* does not exist, unless **−−if−exists** is specified.

lr−policy−list *router*
Lists the polices on *router*.

NAT COMMANDS

[**−−may−exist**] [**−−stateless**]**lr−nat−add** *router type external_ip logical_ip* [*logical_port external_mac*]
Adds the specified NAT to *router*. The *type* must be one of **snat**, **dnat**, or **dnat_and_snat**. The *external_ip* is an IPv4 address. The *logical_ip* is an IPv4 network (e.g 192.168.1.0/24) or an IPv4 address. The *logical_port* and *external_mac* are only accepted when *router* is a distributed router (rather than a gateway router) and *type* is **dnat_and_snat**. The *logical_port* is the name of an existing logical switch port where the *logical_ip* resides. The *external_mac* is an Ethernet address. The *−−stateless*

When **−−stateless** is specified then it implies that we will be not use connection tracker, i.e internal ip and external ip are 1:1 mapped. This implies that **−−stateless** is applicable only to dnat_and_snat type NAT rules. An external ip with **−−stateless** NAT cannot be shared with any other 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*.

When the *logical_port* and *external_mac* are specified, the NAT rule will be programmed on the chassis where the *logical_port* resides. This includes ARP replies for the *external_ip*, which return the value of *external_mac*. All packets transmitted with source IP address equal to *external_ip* will be sent using the *external_mac*.

It is an error if a NAT already exists with the same values of *router*, *type*, *external_ip*, and *logical_ip*, unless **−−may−exist** is specified. When **−−may−exist**, *logical_port*, and *external_mac* are all specified, the existing values of *logical_port* and *external_mac* are overwritten.

[**−−if−exists**] **lr−nat−del** *router* [*type* [*ip*]]
Deletes NATs from *router*. If only *router* is supplied, all the NATs from the logical router are deleted. If *type* is also specified, then all the NATs that match the *type* will be deleted from the logical router. If all the fields are given, then a single NAT rule that matches all the fields will be deleted. When *type* is **snat**, the *ip* should be logical_ip. When *type* is **dnat** or **dnat_and_snat**, the *ip* shoud be external_ip.

It is an error if *ip* is specified and there is no matching NAT entry, unless **−−if−exists** is specified.

lr−nat−list *router*
Lists the NATs on *router*.

LOAD BALANCER COMMANDS

[**−−may−exist** | **−−add−duplicate** | **−−reject** | **−−event**] **lb−add** *lb vip ips* [*protocol*]

Creates a new load balancer named *lb* with the provided *vip* and *ips* or adds the *vip* to an existing *lb*. *vip* should be a virtual IP address (or an IP address and a port number with **:** as a separator). Examples for *vip* are **192.168.1.4**, **fd0f::1**, and **192.168.1.5:8080**. *ips* should be comma separated IP endpoints (or comma separated IP addresses and port numbers with **:** as a separator). *ips* must be the same address family as *vip*. Examples for *ips* are **10.0.0.1,10.0.0.2**or **[fdef::1]:8800,[fdef::2]:8800**.

The optional argument *protocol* must be either **tcp**, **udp** or **sctp**. This argument is useful when a port number is provided as part of the *vip*. If the *protocol* is unspecified and a port number is provided as part of the *vip*, OVN assumes the *protocol* to be **tcp**.

It is an error if the *vip* already exists in the load balancer named *lb*, unless **−−may−exist** is specified. With **−−add−duplicate**, the command really creates a new load balancer with a duplicate name.

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.

If the load balancer is created with **−−event** option and it has no active backends, whenever the lb receives traffic, the event is reported in the Controller_Event table in the SB db. Please note **−−event** option can't be specified with **−−reject** one.

The following example adds a load balancer.

lb−add lb0 30.0.0.10:80 192.168.10.10:80,192.168.10.20:80,192.168.10.30:80 udp

[**−−if−exists**] **lb−del** *lb* [*vip*]

Deletes *lb* or the *vip* from *lb*. If *vip* is supplied, only the *vip* will be deleted from the *lb*. If only the *lb* is supplied, the *lb* will be deleted. It is an error if *vip* does not already exist in *lb*, unless **−−if−exists** is specified.

lb−list [*lb*]

Lists the LBs. If *lb* is also specified, then only the specified *lb* will be listed.

[**−−may−exist**] **ls−lb−add** *switch lb*

Adds the specified *lb* to *switch*. It is an error if a load balancer named *lb* already exists in the *switch*, unless **−−may−exist** is specified.

[**−−if−exists**] **ls−lb−del** *switch* [*lb*]

Removes *lb* from *switch*. If only *switch* is supplied, all the LBs from the logical switch are removed. If *lb* is also specified, then only the *lb* will be removed from the logical switch. It is an error if *lb* does not exist in the *switch*, unless **−−if−exists** is specified.

ls−lb−list *switch*

Lists the LBs for the given *switch*.

[**−−may−exist**] **lr−lb−add** *router lb*

Adds the specified *lb* to *router*. It is an error if a load balancer named *lb* already exists in the *router*, unless **−−may−exist** is specified.

[**−−if−exists**] **lr−lb−del** *router* [*lb*]

Removes *lb* from *router*. If only *router* is supplied, all the LBs from the logical router are removed. If *lb* is also specified, then only the *lb* will be removed from the logical router. It is an error if *lb* does not exist in the *router*, unless **−−if−exists** is specified.

lr−lb−list *router*

Lists the LBs for the given *router*.

DHCP OPTIONS COMMANDS

dhcp−options−create *cidr* [*key*=*value*]
Creates a new DHCP Options entry in the **DHCP_Options** table with the specified **cidr** and optional **external−ids**.

dhcp−options−list
Lists the DHCP Options entries.

dhcp−options−del *dhcp-option*
Deletes the DHCP Options entry referred by *dhcp-option* UUID.

dhcp−options−set−options *dhcp-option* [*key*=*value*]...
Set the DHCP Options for the *dhcp-option* UUID.

dhcp−options−get−options *dhcp-option*
Lists the DHCP Options for the *dhcp-option* UUID.

PORT GROUP COMMANDS

pg−add *group* [*port*]...
Creates a new port group in the **Port_Group** table named **group** with optional **ports** added to the group.

pg−set−ports *group port...*
Sets **ports** on the port group named **group**. It is an error if **group** does not exist.

pg−del *group*
Deletes port group **group**. It is an error if **group** does not exist.

HA CHASSIS GROUP COMMANDS

ha−chassis−group−add *group*
Creates a new HA chassis group in the **HA_Chassis_Group** table named **group**.

ha−chassis−group−del *group*
Deletes the HA chassis group **group**. It is an error if **group** does not exist.

ha−chassis−group−list
Lists the HA chassis group **group** along with the **HA chassis** if any associated with it.

ha−chassis−group−add−chassis *group chassis priority*
Adds a new HA chassis **chassis** to the HA Chassis group **group** with the specified priority. If the **chassis** already exists, then the **priority** is updated. The **chassis** should be the name of the chassis in the **OVN_Southbound**.

ha−chassis−group−remove−chassis *group chassis*
Removes the HA chassis **chassis** from the HA chassis group **group**. It is an error if **chassis** does not exist.

DATABASE COMMANDS

These commands query and modify the contents of **ovsdb** tables. They are a slight abstraction of the **ovsdb** interface and as such they operate at a lower level than other **ovn−nbctl** commands.

Identifying Tables, Records, and Columns

Each of these commands has a *table* parameter to identify a table within the database. Many of them also take a *record* parameter that identifies a particular record within a table. The *record* parameter may be the UUID for a record, which may be abbreviated to its first 4 (or more) hex digits, as long as that is unique. Many tables offer additional ways to identify records. Some commands also take *column* parameters that identify a particular field within the records in a table.

For a list of tables and their columns, see **ovn−nb**(5) or see the table listing from the **−−help** option.

Record names must be specified in full and with correct capitalization, except that UUIDs may be abbreviated to their first 4 (or more) hex digits, as long as that is unique within the table. Names of tables and columns are not case-sensitive, and **−** and **_** are treated interchangeably. Unique abbreviations of table and column names are acceptable, e.g. **d** or **dhcp** is sufficient to identify the **DHCP_Options** table.

Database Values

Each column in the database accepts a fixed type of data. The currently defined basic types, and their representations, are:

integer A decimal integer in the range −2**63 to 2**63−1, inclusive.

real A floating-point number.

Boolean
True or false, written **true** or **false**, respectively.

string An arbitrary Unicode string, except that null bytes are not allowed. Quotes are optional for most strings that begin with an English letter or underscore and consist only of letters, underscores, hyphens, and periods. However, **true** and **false** and strings that match the syntax of UUIDs (see below) must be enclosed in double quotes to distinguish them from other basic types. When double quotes are used, the syntax is that of strings in JSON, e.g. backslashes may be used to escape special characters. The empty string must be represented as a pair of double quotes (**""**).

UUID Either a universally unique identifier in the style of RFC 4122, e.g. **f81d4fae−7dec−11d0−a765−00a0c91e6bf6**, or an @*name* defined by a **get** or **create** command within the same **ovs−vsctl** invocation.

Multiple values in a single column may be separated by spaces or a single comma. When multiple values are present, duplicates are not allowed, and order is not important. Conversely, some database columns can have an empty set of values, represented as **[]**, and square brackets may optionally enclose other non-empty sets or single values as well.

A few database columns are ‘‘maps’’ of key-value pairs, where the key and the value are each some fixed database type. These are specified in the form *key***=***value*, where *key* and *value* follow the syntax for the column’s key type and value type, respectively. When multiple pairs are present (separated by spaces or a comma), duplicate keys are not allowed, and again the order is not important. Duplicate values are allowed. An empty map is represented as **{}**. Curly braces may optionally enclose non-empty maps as well (but use quotes to prevent the shell from expanding **other−config={0=x,1=y}** into **other−config=0=x other−config=1=y**, which may not have the desired effect).

Database Command Syntax

[**−−if−exists**] [**−−columns=***column*[**,***column*]...] **list** *table* [*record*]...
Lists the data in each specified *record*. If no records are specified, lists all the records in *table*.

If **−−columns** is specified, only the requested columns are listed, in the specified order. Otherwise, all columns are listed, in alphabetical order by column name.

Without **−−if−exists**, it is an error if any specified *record* does not exist. With **−−if−exists**, the command ignores any *record* that does not exist, without producing any output.

[**−−columns=***column*[**,***column*]...] **find** *table* [*column*[**:***key*]**=***value*]...
Lists the data in each record in *table* whose *column* equals *value* or, if *key* is specified, whose *column* contains a *key* with the specified *value*. The following operators may be used where **=** is written in the syntax summary:

= != < > <= >=
Selects records in which *column*[**:***key*] equals, does not equal, is less than, is greater than, is less than or equal to, or is greater than or equal to *value*, respectively.

Consider *column*[**:***key*] and *value* as sets of elements. Identical sets are considered equal. Otherwise, if the sets have different numbers of elements, then the set with more elements is considered to be larger. Otherwise, consider a element

from each set pairwise, in increasing order within each set. The first pair that differs determines the result. (For a column that contains key-value pairs, first all the keys are compared, and values are considered only if the two sets contain identical keys.)

{=} {!=}
Test for set equality or inequality, respectively.

{<=} Selects records in which *column*[**:***key*] is a subset of *value*. For example, **flood−vlans{<=}1,2** selects records in which the **flood−vlans** column is the empty set or contains 1 or 2 or both.

{<} Selects records in which *column*[**:***key*] is a proper subset of *value*. For example, **flood−vlans{<}1,2** selects records in which the **flood−vlans** column is the empty set or contains 1 or 2 but not both.

{>=} {>}
Same as **{<=}** and **{<}**, respectively, except that the relationship is reversed. For example, **flood−vlans{>=}1,2** selects records in which the **flood−vlans** column contains both 1 and 2.

The following operators are available only in Open vSwitch 2.16 and later:

{in} Selects records in which every element in *column*[**:***key*] is also in *value*. (This is the same as **{<=}**.)

{not−in}
Selects records in which every element in *column*[**:***key*] is not in *value*.

For arithmetic operators (**= != < > <= >=**), when *key* is specified but a particular record's *column* does not contain *key*, the record is always omitted from the results. Thus, the condition **other−config:mtu!=1500** matches records that have a **mtu** key whose value is not 1500, but not those that lack an **mtu** key.

For the set operators, when *key* is specified but a particular record's *column* does not contain *key*, the comparison is done against an empty set. Thus, the condition **other−config:mtu{!=}1500** matches records that have a **mtu** key whose value is not 1500 and those that lack an **mtu** key.

Don't forget to escape **<** or **>** from interpretation by the shell.

If **−−columns** is specified, only the requested columns are listed, in the specified order. Otherwise all columns are listed, in alphabetical order by column name.

The UUIDs shown for rows created in the same **ovs−vsctl** invocation will be wrong.

[**−−if−exists**] [**−−id=@***name*] **get** *table record* [*column*[**:***key*]]...
Prints the value of each specified *column* in the given *record* in *table*. For map columns, a *key* may optionally be specified, in which case the value associated with *key* in the column is printed, instead of the entire map.

Without **−−if−exists**, it is an error if *record* does not exist or *key* is specified, if *key* does not exist in *record*. With **−−if−exists**, a missing *record* yields no output and a missing *key* prints a blank line.

If **@***name* is specified, then the UUID for *record* may be referred to by that name later in the same **ovs−vsctl** invocation in contexts where a UUID is expected.

Both **−−id** and the *column* arguments are optional, but usually at least one or the other should be specified. If both are omitted, then **get** has no effect except to verify that *record* exists in *table*.

−−id and **−−if−exists** cannot be used together.

[**--if-exists**] **set** *table record column*[**:***key*]**=***value*...
Sets the value of each specified *column* in the given *record* in *table* to *value*. For map columns, a *key* may optionally be specified, in which case the value associated with *key* in that column is changed (or added, if none exists), instead of the entire map.

Without **--if-exists**, it is an error if *record* does not exist. With **--if-exists**, this command does nothing if *record* does not exist.

[**--if-exists**] **add** *table record column* [*key***=**]*value*...
Adds the specified value or key-value pair to *column* in *record* in *table*. If *column* is a map, then *key* is required, otherwise it is prohibited. If *key* already exists in a map column, then the current *value* is not replaced (use the **set** command to replace an existing value).

Without **--if-exists**, it is an error if *record* does not exist. With **--if-exists**, this command does nothing if *record* does not exist.

[**--if-exists**] **remove** *table record column value*...

[**--if-exists**] **remove** *table record column key*...

[**--if-exists**] **remove** *table record column key***=***value*... Removes the specified values or key-value pairs from *column* in *record* in *table*. The first form applies to columns that are not maps: each specified *value* is removed from the column. The second and third forms apply to map columns: if only a *key* is specified, then any key-value pair with the given *key* is removed, regardless of its value; if a *value* is given then a pair is removed only if both key and value match.

It is not an error if the column does not contain the specified key or value or pair.

Without **--if-exists**, it is an error if *record* does not exist. With **--if-exists**, this command does nothing if *record* does not exist.

[**--if-exists**] **clear** *table record column*...
Sets each *column* in *record* in *table* to the empty set or empty map, as appropriate. This command applies only to columns that are allowed to be empty.

Without **--if-exists**, it is an error if *record* does not exist. With **--if-exists**, this command does nothing if *record* does not exist.

[**--id=@***name*] **create** *table column*[**:***key*]**=***value*...
Creates a new record in *table* and sets the initial values of each *column*. Columns not explicitly set will receive their default values. Outputs the UUID of the new row.

If **@***name* is specified, then the UUID for the new row may be referred to by that name elsewhere in the same ***(PN** invocation in contexts where a UUID is expected. Such references may precede or follow the **create** command.

Caution (ovs-vsctl as example)
Records in the Open vSwitch database are significant only when they can be reached directly or indirectly from the **Open_vSwitch** table. Except for records in the **QoS** or **Queue** tables, records that are not reachable from the **Open_vSwitch** table are automatically deleted from the database. This deletion happens immediately, without waiting for additional **ovs-vsctl** commands or other database activity. Thus, a **create** command must generally be accompanied by additional commands *within the same* **ovs-vsctl** *invocation* to add a chain of references to the newly created record from the top-level **Open_vSwitch** record. The **EXAMPLES** section gives some examples that show how to do this.

[**--if-exists**] **destroy** *table record*...
Deletes each specified *record* from *table*. Unless **--if-exists** is specified, each *record*s must exist.

−−all destroy *table*
Deletes all records from the *table*.

Caution (ovs-vsctl as example)
The **destroy** command is only useful for records in the **QoS** or **Queue** tables. Records in other tables are automatically deleted from the database when they become unreachable from the **Open_vSwitch** table. This means that deleting the last reference to a record is sufficient for deleting the record itself. For records in these tables, **destroy** is silently ignored. See the **EXAMPLES** section below for more information.

wait−until *table record* [*column*[**:***key*]**=***value*]...
Waits until *table* contains a record named *record* whose *column* equals *value* or, if *key* is specified, whose *column* contains a *key* with the specified *value*. This command supports the same operators and semantics described for the **find** command above.

If no *column*[**:***key*]**=***value* arguments are given, this command waits only until *record* exists. If more than one such argument is given, the command waits until all of them are satisfied.

Caution (ovs-vsctl as example)
Usually **wait−until** should be placed at the beginning of a set of **ovs−vsctl** commands. For example, **wait−until bridge br0 −− get bridge br0 datapath_id** waits until a bridge named **br0** is created, then prints its **datapath_id** column, whereas **get bridge br0 datapath_id −− wait−until bridge br0** will abort if no bridge named **br0** exists when **ovs−vsctl** initially connects to the database.

Consider specifying **−−timeout=0** along with **−−wait−until**, to prevent **ovs−vsctl** from terminating after waiting only at most 5 seconds.

comment [*arg*]...
This command has no effect on behavior, but any database log record created by the command will include the command and its arguments.

SYNCHRONIZATION COMMANDS

sync Ordinarily, **−−wait=sb** or **−−wait=hv** only waits for changes by the current **ovn−nbctl** invocation to take effect. This means that, if none of the commands supplied to **ovn−nbctl** change the database, then the command does not wait at all. With the **sync** command, however, **ovn−nbctl** waits even for earlier changes to the database to propagate down to the southbound database or all of the OVN chassis, according to the argument to **−−wait**.

REMOTE CONNECTIVITY COMMANDS

get−connection
Prints the configured connection(s).

del−connection
Deletes the configured connection(s).

[**−−inactivity−probe=***msecs*] **set−connection** *target*...
Sets the configured manager target or targets. Use **−−inactivity−probe=***msecs* to override the default idle connection inactivity probe time. Use 0 to disable inactivity probes.

SSL CONFIGURATION COMMANDS

get−ssl Prints the SSL configuration.

del−ssl Deletes the current SSL configuration.

[**−−bootstrap**] **set−ssl** *private-key certificate ca-cert* [*ssl-protocol-list* [*ssl-cipher-list*]]
Sets the SSL configuration.

DAEMON MODE

When it is invoked in the most ordinary way, **ovn−nbctl** connects to an OVSDB server that hosts the northbound database, retrieves a partial copy of the database that is complete enough to do its work, sends a

transaction request to the server, and receives and processes the server's reply. In common interactive use, this is fine, but if the database is large, the step in which **ovn−nbctl** retrieves a partial copy of the database can take a long time, which yields poor performance overall.

To improve performance in such a case, **ovn−nbctl** offers a "daemon mode," in which the user first starts **ovn−nbctl** running in the background and afterward uses the daemon to execute operations. Over several **ovn−nbctl** command invocations, this performs better overall because it retrieves a copy of the database only once at the beginning, not once per program run.

Use the **−−detach** option to start an **ovn−nbctl** daemon. With this option, **ovn−nbctl** prints the name of a control socket to stdout. The client should save this name in environment variable **OVN_NB_DAEMON**. Under the Bourne shell this might be done like this:

```
export OVN_NB_DAEMON=$(ovn-nbctl --pidfile --detach)
```

When **OVN_NB_DAEMON** is set, **ovn−nbctl** automatically and transparently uses the daemon to execute its commands.

When the daemon is no longer needed, kill it and unset the environment variable, e.g.:

```
kill $(cat $OVN_RUNDIR/ovn-nbctl.pid)
unset OVN_NB_DAEMON
```

When using daemon mode, an alternative to the OVN_NB_DAEMON environment variable is to specify a path for the Unix socket. When starting the ovn-nbctl daemon, specify the **−u** option with a full path to the location of the socket file. Here is an exmple:

```
ovn-nbctl --detach -u /tmp/mysock.ctl
```

Then to connect to the running daemon, use the **−u** option with the full path to the socket created when the daemon was started:

```
ovn-nbctl -u /tmp/mysock.ctl show
```

Daemon mode is experimental.

Daemon Commands

Daemon mode is internally implemented using the same mechanism used by **ovs−appctl**. One may also use **ovs−appctl** directly with the following commands:

run [*options*] *command* [*arg*...] [**−−** [*options*] *command* [*arg*...] ...]
: Instructs the daemon process to run one or more **ovn−nbctl** commands described above and reply with the results of running these commands. Accepts the **−−no−wait**, **−−wait**, **−−timeout**, **−−dry−run**, **−−oneline**, and the options described under **Table Formatting Options** in addition to the the command-specific options.

exit
: Causes **ovn−nbctl** to gracefully terminate.

OPTIONS

−−no−wait | −−wait=none
−−wait=sb
−−wait=hv
: These options control whether and how **ovn−nbctl** waits for the OVN system to become up-to-date with changes made in an **ovn−nbctl** invocation.

  By default, or if **−−no−wait** or **−−wait=none**, **ovn−nbctl** exits immediately after confirming that changes have been committed to the northbound database, without waiting.

  With **−−wait=sb**, before **ovn−nbctl** exits, it waits for **ovn−northd** to bring the southbound database up-to-date with the northbound database updates.

With **−−wait=hv**, before **ovn−nbctl** exits, it additionally waits for all OVN chassis (hypervisors and gateways) to become up-to-date with the northbound database updates. (This can become an indefinite wait if any chassis is malfunctioning.)

Ordinarily, **−−wait=sb** or **−−wait=hv** only waits for changes by the current **ovn−nbctl** invocation to take effect. This means that, if none of the commands supplied to **ovn−nbctl** change the database, then the command does not wait at all. Use the **sync** command to override this behavior.

−−print−wait−time

When **−−wait** is specified, the option **−−print−wait−time** can be used to print the time spent on waiting, depending on the value specified in **−−wait** option. If **−−wait=sb** is specified, it prints "ovn-northd delay before processing", which is the time between the Northbound DB update by the command and the moment when **ovn−northd** starts processing the update, and "ovn-northd completion", which is the time between the Northbound DB update and the moment when **ovn−northd** completes the Southbound DB updating successfully. If **−−wait=hv** is specified, in addition to the above information, it also prints "ovn-controller(s) completion", which is the time between the Northbound DB update and the moment when the slowest hypervisor finishes processing the update.

−−db *database*

The OVSDB database remote to contact. If the **OVN_NB_DB** environment variable is set, its value is used as the default. Otherwise, the default is **unix:/ovnnb_db.sock**, but this default is unlikely to be useful outside of single-machine OVN test environments.

−−leader−only

−−no−leader−only

By default, or with **−−leader−only**, when the database server is a clustered database, **ovn−nbctl** will avoid servers other than the cluster leader. This ensures that any data that **ovn−nbctl** reads and reports is up-to-date. With **−−no−leader−only**, **ovn−nbctl** will use any server in the cluster, which means that for read-only transactions it can report and act on stale data (transactions that modify the database are always serialized even with **−−no−leader−only**). Refer to **Understanding Cluster Consistency** in **ovsdb**(7) for more information.

−−shuffle−remotes

−−no−shuffle−remotes

By default, or with **−−shuffle−remotes**, when there are multiple remotes specified in the OVSDB connection string specified by **−−db** or the **OVN_NB_DB** environment variable, the order of the remotes will be shuffled before the client tries to connect. The remotes will be shuffled only once to a new order before the first connection attempt. The following retries, if any, will follow the same new order. The default behavior is to make sure clients of a clustered database can distribute evenly to all memembers of the cluster. With **−−no−shuffle−remotes**, **ovn−nbctl** will use the original order specified in the connection string to connect. This allows user to specify the preferred order, which is particularly useful for testing.

OVN_NBCTL_OPTIONS

User can set one or more **OVN_NBCTL_OPTIONS** options in environment variable. Under the Bourne shell this might be done like this:

```
OVN_NBCTL_OPTIONS="--db=unix:nb1.ovsdb --no-leader-only"
```

When **OVN_NBCTL_OPTIONS** is set, **ovn−nbctl** automatically and transparently uses the environment variable to execute its commands. However user can still over-ride environment options by passing different in cli.

When the environment variable is no longer needed, unset it, e.g.:

```
unset OVN_NBCTL_OPTIONS
```

Daemon Options

−−pidfile[**=***pidfile*]

Causes a file (by default, *program***.pid**) to be created indicating the PID of the running process. If the *pidfile* argument is not specified, or if it does not begin with **/**, then it is created in .

If **−−pidfile** is not specified, no pidfile is created.

−−overwrite−pidfile

By default, when **−−pidfile** is specified and the specified pidfile already exists and is locked by a running process, the daemon refuses to start. Specify **−−overwrite−pidfile** to cause it to instead overwrite the pidfile.

When **−−pidfile** is not specified, this option has no effect.

−−detach

Runs this program as a background process. The process forks, and in the child it starts a new session, closes the standard file descriptors (which has the side effect of disabling logging to the console), and changes its current directory to the root (unless **−−no−chdir** is specified). After the child completes its initialization, the parent exits.

−−monitor

Creates an additional process to monitor this program. If it dies due to a signal that indicates a programming error (**SIGABRT**, **SIGALRM**, **SIGBUS**, **SIGFPE**, **SIGILL**, **SIGPIPE**, **SIGSEGV**, **SIGXCPU**, or **SIGXFSZ**) then the monitor process starts a new copy of it. If the daemon dies or exits for another reason, the monitor process exits.

This option is normally used with **−−detach**, but it also functions without it.

−−no−chdir

By default, when **−−detach** is specified, the daemon changes its current working directory to the root directory after it detaches. Otherwise, invoking the daemon from a carelessly chosen directory would prevent the administrator from unmounting the file system that holds that directory.

Specifying **−−no−chdir** suppresses this behavior, preventing the daemon from changing its current working directory. This may be useful for collecting core files, since it is common behavior to write core dumps into the current working directory and the root directory is not a good directory to use.

This option has no effect when **−−detach** is not specified.

−−no−self−confinement

By default this daemon will try to self-confine itself to work with files under well-known directories determined at build time. It is better to stick with this default behavior and not to use this flag unless some other Access Control is used to confine daemon. Note that in contrast to other access control implementations that are typically enforced from kernel-space (e.g. DAC or MAC), self-confinement is imposed from the user-space daemon itself and hence should not be considered as a full confinement strategy, but instead should be viewed as an additional layer of security.

−−user=*user***:***group*

Causes this program to run as a different user specified in *user***:***group*, thus dropping most of the root privileges. Short forms *user* and **:***group* are also allowed, with current user or group assumed, respectively. Only daemons started by the root user accepts this argument.

On Linux, daemons will be granted **CAP_IPC_LOCK** and **CAP_NET_BIND_SERVICES** before dropping root privileges. Daemons that interact with a datapath, such as **ovs−vswitchd**, will be granted three additional capabilities, namely **CAP_NET_ADMIN**, **CAP_NET_BROADCAST** and **CAP_NET_RAW**. The capability change will apply even if the new user is root.

On Windows, this option is not currently supported. For security reasons, specifying this option will cause the daemon process not to start.

LOGGING OPTIONS

−v[*spec*]
−−verbose=[*spec*]
Sets logging levels. Without any *spec*, sets the log level for every module and destination to **dbg**. Otherwise, *spec* is a list of words separated by spaces or commas or colons, up to one from each category below:

- A valid module name, as displayed by the **vlog/list** command on **ovs−appctl**(8), limits the log level change to the specified module.
- **syslog**, **console**, or **file**, to limit the log level change to only to the system log, to the console, or to a file, respectively. (If **−−detach** is specified, the daemon closes its standard file descriptors, so logging to the console will have no effect.)

  On Windows platform, **syslog** is accepted as a word and is only useful along with the **−−syslog−target** option (the word has no effect otherwise).
- **off**, **emer**, **err**, **warn**, **info**, or **dbg**, to control the log level. Messages of the given severity or higher will be logged, and messages of lower severity will be filtered out. **off** filters out all messages. See **ovs−appctl**(8) for a definition of each log level.

Case is not significant within *spec*.

Regardless of the log levels set for **file**, logging to a file will not take place unless **−−log−file** is also specified (see below).

For compatibility with older versions of OVS, **any** is accepted as a word but has no effect.

−v
−−verbose
Sets the maximum logging verbosity level, equivalent to **−−verbose=dbg**.

−vPATTERN:*destination***:***pattern*
−−verbose=PATTERN:*destination***:***pattern*
Sets the log pattern for *destination* to *pattern*. Refer to **ovs−appctl**(8) for a description of the valid syntax for *pattern*.

−vFACILITY:*facility*
−−verbose=FACILITY:*facility*
Sets the RFC5424 facility of the log message. *facility* can be one of **kern**, **user**, **mail**, **daemon**, **auth**, **syslog**, **lpr**, **news**, **uucp**, **clock**, **ftp**, **ntp**, **audit**, **alert**, **clock2**, **local0**, **local1**, **local2**, **local3**, **local4**, **local5**, **local6** or **local7**. If this option is not specified, **daemon** is used as the default for the local system syslog and **local0** is used while sending a message to the target provided via the **−−syslog−target** option.

−−log−file[**=***file*]
Enables logging to a file. If *file* is specified, then it is used as the exact name for the log file. The default log file name used if *file* is omitted is **/usr/local/var/log/ovn/***program***.log**.

−−syslog−target=*host***:***port*
Send syslog messages to UDP *port* on *host*, in addition to the system syslog. The *host* must be a numerical IP address, not a hostname.

−−syslog−method=*method*
Specify *method* as how syslog messages should be sent to syslog daemon. The following forms are supported:

- **libc**, to use the libc **syslog()** function. Downside of using this options is that libc adds fixed prefix to every message before it is actually sent to the syslog daemon over **/dev/log** UNIX domain socket.
- **unix:***file*, to use a UNIX domain socket directly. It is possible to specify arbitrary message format with this option. However, **rsyslogd 8.9** and older versions use hard coded

parser function anyway that limits UNIX domain socket use. If you want to use arbitrary message format with older **rsyslogd** versions, then use UDP socket to localhost IP address instead.

- **udp:***ip***:***port*, to use a UDP socket. With this method it is possible to use arbitrary message format also with older **rsyslogd**. When sending syslog messages over UDP socket extra precaution needs to be taken into account, for example, syslog daemon needs to be configured to listen on the specified UDP port, accidental iptables rules could be interfering with local syslog traffic and there are some security considerations that apply to UDP sockets, but do not apply to UNIX domain sockets.
- **null**, to discard all messages logged to syslog.

The default is taken from the **OVS_SYSLOG_METHOD** environment variable; if it is unset, the default is **libc**.

TABLE FORMATTING OPTIONS

These options control the format of output from the **list** and **find** commands.

–f *format*
––format=*format*

Sets the type of table formatting. The following types of *format* are available:

table 2-D text tables with aligned columns.

list (default)
A list with one column per line and rows separated by a blank line.

html HTML tables.

csv Comma-separated values as defined in RFC 4180.

json JSON format as defined in RFC 4627. The output is a sequence of JSON objects, each of which corresponds to one table. Each JSON object has the following members with the noted values:

caption
The table's caption. This member is omitted if the table has no caption.

headings
An array with one element per table column. Each array element is a string giving the corresponding column's heading.

data An array with one element per table row. Each element is also an array with one element per table column. The elements of this second-level array are the cells that constitute the table. Cells that represent OVSDB data or data types are expressed in the format described in the OVSDB specification; other cells are simply expressed as text strings.

–d *format*
––data=*format*

Sets the formatting for cells within output tables unless the table format is set to **json**, in which case **json** formatting is always used when formatting cells. The following types of *format* are available:

string (default)
The simple format described in the **Database Values** section of **ovs–vsctl**(8).

bare The simple format with punctuation stripped off: **[]** and **{}** are omitted around sets, maps, and empty columns, items within sets and maps are space-separated, and strings are never quoted. This format may be easier for scripts to parse.

json The RFC 4627 JSON format as described above.

−−no−headings
This option suppresses the heading row that otherwise appears in the first row of table output.

−−pretty
By default, JSON in output is printed as compactly as possible. This option causes JSON in output to be printed in a more readable fashion. Members of objects and elements of arrays are printed one per line, with indentation.

This option does not affect JSON in tables, which is always printed compactly.

−−bare Equivalent to **−−format=list −−data=bare −−no−headings**.

PKI Options

PKI configuration is required to use SSL for the connection to the database.

−p *privkey.pem*
−−private−key=*privkey.pem*
Specifies a PEM file containing the private key used as identity for outgoing SSL connections.

−c *cert.pem*
−−certificate=*cert.pem*
Specifies a PEM file containing a certificate that certifies the private key specified on **−p** or **−−private−key** to be trustworthy. The certificate must be signed by the certificate authority (CA) that the peer in SSL connections will use to verify it.

−C *cacert.pem*
−−ca−cert=*cacert.pem*
Specifies a PEM file containing the CA certificate for verifying certificates presented to this program by SSL peers. (This may be the same certificate that SSL peers use to verify the certificate specified on **−c** or **−−certificate**, or it may be a different one, depending on the PKI design in use.)

−C none
−−ca−cert=none
Disables verification of certificates presented by SSL peers. This introduces a security risk, because it means that certificates cannot be verified to be those of known trusted hosts.

−−bootstrap−ca−cert=*cacert.pem*
When *cacert.pem* exists, this option has the same effect as **−C** or **−−ca−cert**. If it does not exist, then the executable will attempt to obtain the CA certificate from the SSL peer 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, but it may be useful for bootstrapping.

This option is only useful if the SSL peer sends its CA certificate as part of the SSL certificate chain. The SSL protocol does not require the server to send the CA certificate.

This option is mutually exclusive with **−C** and **−−ca−cert**.

Other Options

−h
−−help Prints a brief help message to the console.

−V
−−version
Prints version information to the console.