NAME
vtep−ctl − utility for querying and configuring a VTEP database

SYNOPSIS
vtep−ctl [*options*] −− [*options*] *command* [*args*] [−− [*options*] *command* [*args*]]...

DESCRIPTION
The **vtep−ctl** program configures a VTEP database. See **vtep**(5) for comprehensive documentation of the database schema.

vtep−ctl connects to an **ovsdb−server** process that maintains a VTEP configuration database. Using this connection, it queries and possibly applies changes to the database, depending on the supplied commands.

vtep−ctl can perform any number of commands in a single run, implemented as a single atomic transaction against the database.

The **vtep−ctl** command line begins with global options (see **OPTIONS** below for details). The global options are followed by one or more commands. Each command should begin with −− by itself as a command-line argument, to separate it from the following commands. (The −− before the first command is optional.) The command itself starts with command-specific options, if any, followed by the command name and any arguments. See **EXAMPLES** below for syntax examples.

OPTIONS
The following options affect the behavior **vtep−ctl** as a whole. Some individual commands also accept their own options, which are given just before the command name. If the first command on the command line has options, then those options must be separated from the global options by −−.

−−**db=***server*

> Sets *server* as the database server that **vtep−ctl** contacts to query or modify configuration. The default is **unix:/usr/local/var/run/openvswitch/db.sock**. *server* must take one of the following forms:

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) in IPv4 or IPv6 address format. If *ip* is an IPv6 address, then wrap *ip* with square brackets, e.g.: **ssl:[::1]:6640**. The −−**private−key**, −−**certificate**, and −−**ca−cert** options are mandatory when this form is used.

tcp:*ip***:***port*

> Connect to the given TCP *port* on *ip*, where *ip* can be IPv4 or IPv6 address. If *ip* is an IPv6 address, then wrap *ip* with square brackets, e.g.: **tcp:[::1]:6640**.

unix:*file*

> On POSIX, connect to the Unix domain server socket named *file*.

> On Windows, connect to a localhost TCP port whose value is written in *file*.

pssl:*port*[**:***ip*]

> Listen on the given SSL *port* for a connection. By default, connections are not bound to a particular local IP address and it listens only on IPv4 (but not IPv6) addresses, but specifying *ip* limits connections to those from the given *ip*, either IPv4 or IPv6 address. If *ip* is an IPv6 address, then wrap *ip* with square brackets, e.g.: **pssl:6640:[::1]**. The −−**private−key**, −−**certificate**, and −−**ca−cert** options are mandatory when this form is used.

ptcp:*port*[**:***ip*]

> Listen on the given TCP *port* for a connection. By default, connections are not bound to a particular local IP address and it listens only on IPv4 (but not IPv6) addresses, but *ip* may be specified to listen only for connections to the given *ip*, either IPv4 or IPv6 address. If *ip* is an IPv6 address, then wrap *ip* with square brackets, e.g.: **ptcp:6640:[::1]**.

punix:*file*

On POSIX, listen on the Unix domain server socket named *file* for a connection.

On Windows, listen on a kernel chosen TCP port on the localhost. The kernel chosen TCP port value is written in *file*.

−−no−syslog

By default, **vtep−ctl** logs its arguments and the details of any changes that it makes to the system log. This option disables this logging.

This option is equivalent to **−−verbose=vtep_ctl:syslog:warn**.

−−oneline

Modifies the output format so that the output for each command is printed on a single line. New-line characters that would otherwise separate lines are printed as **\n**, and any instances of \ that would otherwise appear in the output are doubled. Prints a blank line for each command that has no output. This option does not affect the formatting of output from the **list** or **find** commands; see **Table Formatting Options** below.

−−dry−run

Prevents **vtep−ctl** from actually modifying the database.

−t *secs*

−−timeout=*secs*

By default, or with a *secs* of **0**, **vtep−ctl** waits forever for a response from the database. This option limits runtime to approximately *secs* seconds. If the timeout expires, **vtep−ctl** will exit with a **SIGALRM** signal. (A timeout would normally happen only if the database cannot be contacted, or if the system is overloaded.)

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 (default)

2-D text tables with aligned columns.

list   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. 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   JSON.

The **json** output format always outputs cells in JSON format, ignoring this option.

−−no−heading

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**.

Public Key Infrastructure Options

−p *privkey.pem*

−−private−key=*privkey.pem*

Specifies a PEM file containing the private key used as **vtep−ctl**'s 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 that **vtep−ctl** should use to verify certificates presented to it 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 **vtep−ctl** 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**.

−−peer−ca−cert=*peer-cacert.pem*

Specifies a PEM file that contains one or more additional certificates to send to SSL peers. *peer-cacert.pem* should be the CA certificate used to sign **vtep−ctl**'s own certificate, that is, the certificate specified on **−c** or **−−certificate**. If **vtep−ctl**'s certificate is self-signed, then **−−certificate** and **−−peer−ca−cert** should specify the same file.

This option is not useful in normal operation, because the SSL peer must already have the CA certificate for the peer to have any confidence in **vtep−ctl**'s identity. However, this offers a way for a new installation to bootstrap the CA certificate on its first SSL connection.

−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, **vtep−ctl** 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/openvswitch/vtep−ctl.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* how syslog messages should be sent to syslog daemon. Following forms are supported:

• **libc**, use libc **syslog()** function. This is the default behavior. 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*, use 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*, use 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.

−**h**

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

−**V**

−−**version**

Prints version information to the console.

COMMANDS

The commands implemented by **vtep−ctl** are described in the sections below.

Physical Switch Commands

These commands examine and manipulate physical switches.

[−−**may−exist**] **add−ps** *pswitch*

Creates a new physical switch named *pswitch*. Initially the switch will have no ports.

Without −−**may−exist**, attempting to create a switch that exists is an error. With −−**may−exist**, this command does nothing if *pswitch* already exists.

[−−**if−exists**] **del−ps** *pswitch*

Deletes *pswitch* and all of its ports.

Without −−**if−exists**, attempting to delete a switch that does not exist is an error. With −−**if−exists**, attempting to delete a switch that does not exist has no effect.

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

ps−exists *pswitch*

Tests whether *pswitch* exists. If so, **vtep−ctl** exits successfully with exit code 0. If not, **vtep−ctl** exits unsuccessfully with exit code 2.

Port Commands

These commands examine and manipulate VTEP physical ports.

list−ports *pswitch*

Lists all of the ports within *pswitch* on standard output, one per line.

[−−**may−exist**] **add−port** *pswitch port*

Creates on *pswitch* a new port named *port* from the network device of the same name.

Without −−**may−exist**, attempting to create a port that exists is an error. With −−**may−exist**, this command does nothing if *port* already exists on *pswitch*.

[−−**if−exists**] **del−port** [*pswitch*] *port*

> Deletes *port*. If *pswitch* is omitted, *port* is removed from whatever switch contains it; if *pswitch* is specified, it must be the switch that contains *port*.

> Without −−**if−exists**, attempting to delete a port that does not exist is an error. With −−**if−exists**, attempting to delete a port that does not exist has no effect.

Logical Switch Commands

These commands examine and manipulate logical switches.

[−−**may−exist**] **add−ls** *lswitch*

> Creates a new logical switch named *lswitch*. Initially the switch will have no locator bindings.

> Without −−**may−exist**, attempting to create a switch that exists is an error. With −−**may−exist**, this command does nothing if *lswitch* already exists.

[−−**if−exists**] **del−ls** *lswitch*

> Deletes *lswitch*.

> Without −−**if−exists**, attempting to delete a switch that does not exist is an error. With −−**if−exists**, attempting to delete a switch that does not exist has no effect.

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

ls−exists *lswitch*

> Tests whether *lswitch* exists. If so, **vtep−ctl** exits successfully with exit code 0. If not, **vtep−ctl** exits unsuccessfully with exit code 2.

bind−ls *pswitch port vlan lswitch*

> Bind logical switch *lswitch* to the *port*/*vlan* combination on the physical switch *pswitch*.

unbind−ls *pswitch port vlan*

> Remove the logical switch binding from the *port*/*vlan* combination on the physical switch *pswitch*.

list−bindings *pswitch port*

> List the logical switch bindings for *port* on the physical switch *pswitch*.

Logical Router Commands

These commands examine and manipulate logical routers.

[−−**may−exist**] **add−lr** *lrouter*

> Creates a new logical router named *lrouter*.

> Without −−**may−exist**, attempting to create a router that exists is an error. With −−**may−exist**, this command does nothing if *lrouter* already exists.

[−−**if−exists**] **del−lr** *lrouter*

> Deletes *lrouter*.

> Without −−**if−exists**, attempting to delete a router that does not exist is an error. With −−**if−exists**, attempting to delete a router that does not exist has no effect.

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

lr−exists *lrouter*

> Tests whether *lrouter* exists. If so, **vtep−ctl** exits successfully with exit code 0. If not, **vtep−ctl** exits unsuccessfully with exit code 2.

Local MAC Binding Commands

These commands examine and manipulate local MAC bindings for the logical switch. The local maps are written by the VTEP to refer to MACs it has learned on its physical ports.

add−ucast−local *lswitch mac* [*encap*] *ip*

> Map the unicast Ethernet address *mac* to the physical location *ip* using encapsulation *encap* on *lswitch*. If *encap* is not specified, the default is "vxlan_over_ipv4". The local mappings are used

by the VTEP to refer to MACs learned on its physical ports.

del−ucast−local *lswitch mac*
>   Remove the local unicast Ethernet address *mac* map from *lswitch*. The local mappings are used by the VTEP to refer to MACs learned on its physical ports.

add−mcast−local *lswitch mac* [*encap*] *ip*
>   Add physical location *ip* using encapsulation *encap* to the local mac binding table for multicast Ethernet address *mac* on *lswitch*. If *encap* is not specified, the default is "vxlan_over_ipv4". The local mappings are used by the VTEP to refer to MACs learned on its physical ports.

del−mcast−local *lswitch mac* [*encap*] *ip*
>   Remove physical location *ip* using encapsulation *encap* from the local mac binding table for multicast Ethernet address *mac* on *lswitch*. If *encap* is not specified, the default is "vxlan_over_ipv4". The local mappings are used by the VTEP to refer to MACs learned on its physical ports.

clear−local−macs *lswitch*
>   Clear the local MAC bindings for *lswitch*.

list−local−macs *lswitch*
>   List the local MAC bindings for *lswitch*, one per line.

Remote MAC Binding Commands

These commands examine and manipulate local and remote MAC bindings for the logical switch. The remote maps are written by the network virtualization controller to refer to MACs that it has learned.

add−ucast−remote *lswitch mac* [*encap*] *ip*
>   Map the unicast Ethernet address *mac* to the physical location *ip* using encapsulation *encap* on *lswitch*. If *encap* is not specified, the default is "vxlan_over_ipv4". The remote mappings are used by the network virtualization platform to refer to MACs that it has learned.

del−ucast−remote *lswitch mac*
>   Remove the remote unicast Ethernet address *mac* map from *lswitch*. The remote mappings are used by the network virtualization platform to refer to MACs that it has learned.

add−mcast−remote *lswitch mac* [*encap*] *ip*
>   Add physical location *ip* using encapsulation *encap* to the remote mac binding table for multicast Ethernet address *mac* on *lswitch*. If *encap* is not specified, the default is "vxlan_over_ipv4". The remote mappings are used by the network virtualization platform to refer to MACs that it has learned.

del−mcast−remote *lswitch mac* [*encap*] *ip*
>   Remove physical location *ip* using encapsulation *encap* from the remote mac binding table for multicast Ethernet address *mac* on *lswitch*. If *encap* is not specified, the default is "vxlan_over_ipv4". The remote mappings are used by the network virtualization platform to refer to MACs that it has learned.

clear−remote−macs *lswitch*
>   Clear the remote MAC bindings for *lswitch*.

list−remote−macs *lswitch*
>   List the remote MAC bindings for *lswitch*, one per line.

Manager Connectivity

These commands manipulate the **managers** column in the **Global** table and rows in the **Managers** table. When **ovsdb−server** is configured to use the **managers** column for OVSDB connections (as described in **INSTALL.Linux** and in the startup scripts provided with Open vSwitch), this allows the administrator to use **vtep−ctl** to configure database connections.

get−manager
>   Prints the configured manager(s).

del−manager
> Deletes the configured manager(s).

set−manager *target*...
> Sets the configured manager target or targets. Each *target* may use any of the following forms:

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) in IPv4 or IPv6 address format. If *ip* is an IPv6 address, then wrap *ip* with square brackets, e.g.: **ssl:[::1]:6640**. The **−−private−key**, **−−certificate**, and **−−ca−cert** options are mandatory when this form is used.

tcp:*ip***:***port*
> Connect to the given TCP *port* on *ip*, where *ip* can be IPv4 or IPv6 address. If *ip* is an IPv6 address, then wrap *ip* with square brackets, e.g.: **tcp:[::1]:6640**.

unix:*file*
> On POSIX, connect to the Unix domain server socket named *file*.
>
> On Windows, connect to a localhost TCP port whose value is written in *file*.

pssl:*port*[**:***ip*]
> Listen on the given SSL *port* for a connection. By default, connections are not bound to a particular local IP address and it listens only on IPv4 (but not IPv6) addresses, but specifying *ip* limits connections to those from the given *ip*, either IPv4 or IPv6 address. If *ip* is an IPv6 address, then wrap *ip* with square brackets, e.g.: **pssl:6640:[::1]**. The **−−private−key**, **−−certificate**, and **−−ca−cert** options are mandatory when this form is used.

ptcp:*port*[**:***ip*]
> Listen on the given TCP *port* for a connection. By default, connections are not bound to a particular local IP address and it listens only on IPv4 (but not IPv6) addresses, but *ip* may be specified to listen only for connections to the given *ip*, either IPv4 or IPv6 address. If *ip* is an IPv6 address, then wrap *ip* with square brackets, e.g.: **ptcp:6640:[::1]**.

punix:*file*
> On POSIX, listen on the Unix domain server socket named *file* for a connection.
>
> On Windows, listen on a kernel chosen TCP port on the localhost. The kernel chosen TCP port value is written in *file*.

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 **vtep−ctl** 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, and 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.

The following tables are currently defined:

Global Top-level configuration for a hardware VTEP. This table contains exactly one record, identified by specifying **.** as the record name.

Manager
> Configuration for an OVSDB connection. Records may be identified by target (e.g. **tcp:1.2.3.4**).

Physical_Switch
> A physical switch that implements a VTEP. Records may be identified by physical switch name.

Physical_Port
> A port within a physical switch.

Logical_Binding_Stats
> Reports statistics for the logical switch with which a VLAN on a physical port is associated.

Logical_Switch
> A logical Ethernet switch. Records may be identified by logical switch name.

Ucast_Macs_Local
> Mapping of locally discovered unicast MAC addresses to tunnels.

Ucast_Macs_Remote
> Mapping of remotely programmed unicast MAC addresses to tunnels.

Mcast_Macs_Local
> Mapping of locally discovered multicast MAC addresses to tunnels.

Mcast_Macs_Remote
> Mapping of remotely programmed multicast MAC addresses to tunnels.

Physical_Locator_Set
> A set of one or more physical locators.

Physical_Locator
> Identifies an endpoint to which logical switch traffic may be encapsulated and forwarded. Records may be identified by physical locator name.

Record names must be specified in full and with correct capitalization. Names of tables and columns are not case-sensitive, and −− and _ are treated interchangeably. Unique abbreviations are acceptable, e.g. **man** or **m** is sufficient to identify the **Manager** 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 **vtep−ctl** 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.

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 **vtep−ctl** 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 **vtep−ctl** 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 **vtep−ctl** invocation in contexts where a UUID is expected. Such references may precede or follow the **create** command.

Caution (ovs-vsctl as exmaple)
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 exmaple)
    :   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*. Any of the operators **!=**, **<**, **>**, **<=**, or **>=** may be substituted for = to test for inequality, less than, greater than, less than or equal to, or greater than or equal to, respectively. (Don't forget to escape **<** or **>** from interpretation by the shell.)

    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 exmaple)
    :   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 data-path_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 **vtep−ctl** 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.

EXIT STATUS

0
:   Successful program execution.

1
:   Usage, syntax, or configuration file error.

2
:   The *switch* argument to **ps−exists** specified the name of a physical switch that does not exist.

SEE ALSO

ovsdb−server(1), **vtep**(5).