## NAME

ovs−vsctl − utility for querying and configuring **ovs−vswitchd**

## SYNOPSIS

**ovs−vsctl** [*options*] **−−** [*options*] *command* [*args*] [**−−** [*options*] *command* [*args*]]...

## DESCRIPTION

The **ovs−vsctl** program configures **ovs−vswitchd**(8) by providing a high−level interface to its configuration database. See **ovs−vswitchd.conf.db**(5) for comprehensive documentation of the database schema.

**ovs−vsctl** connects to an **ovsdb−server** process that maintains an Open vSwitch configuration database. Using this connection, it queries and possibly applies changes to the database, depending on the supplied commands. Then, if it applied any changes, by default it waits until **ovs−vswitchd** has finished reconfiguring itself before it exits. (If you use **ovs−vsctl** when **ovs−vswitchd** is not running, use **−−no−wait**.)

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

The **ovs−vsctl** 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.

### Linux VLAN Bridging Compatibility

The **ovs−vsctl** program supports the model of a bridge implemented by Open vSwitch, in which a single bridge supports ports on multiple VLANs. In this model, each port on a bridge is either a trunk port that potentially passes packets tagged with 802.1Q headers that designate VLANs or it is assigned a single implicit VLAN that is never tagged with an 802.1Q header.

For compatibility with software designed for the Linux bridge, **ovs−vsctl** also supports a model in which traffic associated with a given 802.1Q VLAN is segregated into a separate bridge. A special form of the **add−br** command (see below) creates a "fake bridge" within an Open vSwitch bridge to simulate this behavior. When such a "fake bridge" is active, **ovs−vsctl** will treat it much like a bridge separate from its "parent bridge," but the actual implementation in Open vSwitch uses only a single bridge, with ports on the fake bridge assigned the implicit VLAN of the fake bridge of which they are members. (A fake bridge for VLAN 0 receives packets that have no 802.1Q tag or a tag with VLAN 0.)

## OPTIONS

The following options affect the behavior **ovs−vsctl** 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 **ovs−vsctl** contacts to query or modify configuration. *server* may be an OVSDB active or passive connection method, as described in **ovsdb**(7). The default is **unix:/usr/local/var/run/openvswitch/db.sock**.

**−−no−wait**

Prevents **ovs−vsctl** from waiting for **ovs−vswitchd** to reconfigure itself according to the modified database. This option should be used if **ovs−vswitchd** is not running; otherwise, **ovs−vsctl** will not exit until **ovs−vswitchd** starts.

This option has no effect if the commands specified do not change the database.

**−−no−syslog**

By default, **ovs−vsctl** 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=vsctl: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 **ovs−vsctl** from actually modifying the database.

**−t** *secs*
**−−timeout=***secs*

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

**−−retry**

> Without this option, if **ovs−vsctl** connects outward to the database server (the default) then **ovs−vsctl** will try to connect once and exit with an error if the connection fails (which usually means that **ovsdb−server** is not running).

> With this option, or if **−−db** specifies that **ovs−vsctl** should listen for an incoming connection from the database server, then **ovs−vsctl** will wait for a connection to the database forever.

> Regardless of this setting, **−−timeout** always limits how long **ovs−vsctl** will wait.

**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 below.

**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**.

**−−max−column-width=**_n_
> For table output only, limits the width of any column in the output to *n* columns. Longer cell data is truncated to fit, as necessary. Columns are always wide enough to display the column names, if the heading row is printed.

## Public Key Infrastructure Options

**−p** *privkey.pem*
**−−private−key=***privkey.pem*
> Specifies a PEM file containing the private key used as **ovs−vsctl**'s identity for outgoing SSL/TLS 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/TLS connections will use to verify it.

**−C** *cacert.pem*
**−−ca−cert=***cacert.pem*
> Specifies a PEM file containing the CA certificate that **ovs−vsctl** should use to verify certificates presented to it by SSL/TLS peers. (This may be the same certificate that SSL/TLS 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/TLS 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 **ovs−vsctl** will attempt to obtain the CA certificate from the SSL/TLS peer on its first SSL/TLS 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/TLS connections must be authenticated by a certificate signed by the CA certificate thus obtained.

> **This option exposes the SSL/TLS 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/TLS peer sends its CA certificate as part of the SSL/TLS certificate chain.  SSL/TLS protocols do 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/TLS peers. *peer-cacert.pem* should be the CA certificate used to sign **ovs−vsctl**'s own certificate, that is, the certificate specified on **−c** or **−−certificate**.  If **ovs−vsctl**'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/TLS peer must already have the CA certificate for the peer to have any confidence in **ovs−vsctl**'s identity.  However, this offers a way for a new installation to bootstrap the CA certificate on its first SSL/TLS 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, **ovs−vsctl** 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/ovs−vsctl.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. 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.

> - **null**, discards all messages logged to syslog.

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

**−h**

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

**−V**

**−−version**

> Prints version information to the console.

## COMMANDS

> The commands implemented by **ovs−vsctl** are described in the sections below.

### Open vSwitch Commands

> These commands work with an Open vSwitch as a whole.

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

> Any successful **ovs−vsctl** command automatically initializes the Open vSwitch database if it is empty. This command is provided to initialize the database without executing any other command.

[**−−filter=***filter−rule*[,*filter−rule*]...] **show**

> Prints a brief overview of the database contents. If **−−filter** is specified, output is filtered according to the rules. Each *filter−rule* has the form **table−name**(*filter*[|*filter*]...). The row in this table is shown only if the printed version of it, including all the rows referenced from this one, contains one of the *filter*s as a sub-string.

**emer−reset**

> Reset the configuration into a clean state. It deconfigures OpenFlow controllers, OVSDB servers, and SSL/TLS, and deletes port mirroring, **fail_mode**, NetFlow, sFlow, and IPFIX configuration. This command also removes all **other−config** keys from all database records, except that **other−config:hwaddr** is preserved if it is present in a Bridge record. Other networking configuration is left as-is.

**Bridge Commands**

These commands examine and manipulate Open vSwitch bridges.

[**−−may−exist**] **add−br** *bridge*

Creates a new bridge named *bridge*. Initially the bridge will have no ports (other than *bridge* it-self).

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

[**−−may−exist**] **add−br** *bridge parent vlan*

Creates a "fake bridge" named *bridge* within the existing Open vSwitch bridge *parent*, which must already exist and must not itself be a fake bridge. The new fake bridge will be on 802.1Q VLAN *vlan*, which must be an integer between 0 and 4095. The parent bridge must not already have a fake bridge for *vlan*. Initially *bridge* will have no ports (other than *bridge* itself).

Without **−−may−exist**, attempting to create a bridge that exists is an error. With **−−may−exist**, this command does nothing if *bridge* already exists as a VLAN bridge under *parent* for *vlan*.

[**−−if−exists**] **del−br** *bridge*

Deletes *bridge* and all of its ports. If *bridge* is a real bridge, this command also deletes any fake bridges that were created with *bridge* as parent, including all of their ports.

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

[**−−real**|**−−fake**] **list−br**

Lists all existing real and fake bridges on standard output, one per line. With **−−real** or **−−fake**, only bridges of that type are returned.

**br−exists** *bridge*

Tests whether *bridge* exists as a real or fake bridge. If so, **ovs−vsctl** exits successfully with exit code 0. If not, **ovs−vsctl** exits unsuccessfully with exit code 2.

**br−to−vlan** *bridge*

If *bridge* is a fake bridge, prints the bridge's 802.1Q VLAN as a decimal integer. If *bridge* is a real bridge, prints 0.

**br−to−parent** *bridge*

If *bridge* is a fake bridge, prints the name of its parent bridge. If *bridge* is a real bridge, print *bridge*.

**br−set−external−id** *bridge key* [*value*]

Sets or clears an "external ID" value on *bridge*. These values are intended to identify entities external to Open vSwitch with which *bridge* is associated, e.g. the bridge's identifier in a virtualization management platform. The Open vSwitch database schema specifies well-known *key* values, but *key* and *value* are otherwise arbitrary strings.

If *value* is specified, then *key* is set to *value* for *bridge*, overwriting any previous value. If *value* is omitted, then *key* is removed from *bridge*'s set of external IDs (if it was present).

For real bridges, the effect of this command is similar to that of a **set** or **remove** command in the **external−ids** column of the **Bridge** table. For fake bridges, it actually modifies keys with names prefixed by **fake−bridge−** in the **Port** table.

**br−get−external−id** *bridge* [*key*]

Queries the external IDs on *bridge*. If *key* is specified, the output is the value for that *key* or the empty string if *key* is unset. If *key* is omitted, the output is *key*=*value*, one per line, for each key-value pair.

For real bridges, the effect of this command is similar to that of a **get** command in the **exter-nal−ids** column of the **Bridge** table. For fake bridges, it queries keys with names prefixed by **fake−bridge−** in the **Port** table.

**Port Commands**

These commands examine and manipulate Open vSwitch ports. These commands treat a bonded port as a single entity.

**list−ports** *bridge*

Lists all of the ports within *bridge* on standard output, one per line. The local port *bridge* is not included in the list.

[**−−may−exist**] **add−port** *bridge port* [*column*[**:***key*]=*value*]...

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

Optional arguments set values of column in the Port record created by the command. For example, **tag=9** would make the port an access port for VLAN 9. The syntax is the same as that for the **set** command (see **Database Commands** below).

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 *bridge* and is not a bonded port.

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

Deletes *port*. If *bridge* is omitted, *port* is removed from whatever bridge contains it; if *bridge* is specified, it must be the real or fake bridge 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.

[**−−if−exists**] **−−with−iface del−port** [*bridge*] *iface*

Deletes the port named *iface* or that has an interface named *iface*. If *bridge* is omitted, the port is removed from whatever bridge contains it; if *bridge* is specified, it must be the real or fake bridge that contains the port.

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

**port−to−br** *port*

Prints the name of the bridge that contains *port* on standard output.

**Bond Commands**

These commands work with ports that have more than one interface, which Open vSwitch calls "bonds."

[**−−fake−iface**] **add−bond** *bridge port iface*... [*column*[**:***key*]=*value*]...

Creates on *bridge* a new port named *port* that bonds together the network devices given as each *iface*. At least two interfaces must be named. If the interfaces are DPDK enabled then the transaction will need to include operations to explicitly set the interface type to 'dpdk'.

Optional arguments set values of column in the Port record created by the command. The syntax is the same as that for the **set** command (see **Database Commands** below).

With **−−fake−iface**, a fake interface with the name *port* is created. This should only be used for compatibility with legacy software that requires it.

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 *bridge* and bonds together exactly the specified interfaces.

[**−−may−exist**] **add−bond−iface** *bond iface*

Adds *iface* as a new bond interface to the existing port *bond*. If *bond* previously had only one port, this transforms it into a bond.

Without **−−may−exist**, attempting to add an *iface* that is already part of *bond* is an error. With **−−may−exist**, this command does nothing if *iface* is already part of *bond*. (It is still an error if *iface* is an interface of some other port or bond.)

[**−−if−exists**] **del−bond−iface** [*bond*] *iface*

Removes *iface* from its port. If *bond* is omitted, *iface* is removed from whatever port contains it; if *bond* is specified, it must be the port that contains *bond*.

If removing *iface* causes its port to have only a single interface, then that port transforms from a bond into an ordinary port.  It is an error if *iface* is the only interface in its port.

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

## Interface Commands

These commands examine the interfaces attached to an Open vSwitch bridge.  These commands treat a bonded port as a collection of two or more interfaces, rather than as a single port.

**list−ifaces** *bridge*
> Lists all of the interfaces within *bridge* on standard output, one per line.  The local port *bridge* is not included in the list.

**iface−to−br** *iface*
> Prints the name of the bridge that contains *iface* on standard output.

## Conntrack Zone Commands

These commands query and modify datapath CT zones, Timeout Policies and Limits.

[**−−may−exist**] **add−zone−tp** *datapath* **zone=***zone_id policies*
> Creates a conntrack zone timeout policy with *zone_id* in *datapath*.  The *policies* consist of *key***=***value* pairs, separated by spaces.  For example, **icmp_first=30 icmp_reply=60** specifies a 30-second timeout policy for the first ICMP packet and a 60-second policy for ICMP reply packets.  See the **CT_Timeout_Policy** table in **ovs-vswitchd.conf.db**(5) for the supported keys.
>
> Without **−−may−exist**, attempting to add a *policy* for *zone_id* that already has a policy is an error. With **−−may−exist**, this command does nothing if policy for *zone_id* already exists.

[**−−if−exists**] **del−zone−tp** *datapath* **zone=***zone_id*
> Delete the timeout policy associated with *zone_id* from *datapath*.
>
> Without **−−if−exists**, attempting to delete a policy for zone that does not exist or doesn't have a policy is an error.  With **−−if−exists**, attempting to delete a a policy that does not exist has no effect.

**list−zone−tp** *datapath*
> Prints the timeout policies of all zones in *datapath*.

**set−zone−limit** *datapath zone_id*|**default** *zone_limit*
> Sets a conntrack zone limit with *zone_id*|*default* in *datapath*.  The *limit* with value **0** means unlimited.

[**−−if−exists**] **del−zone−limit** *datapath zone_id*|**default**
> Delete the limit associated with *zone_id* from *datapath*.
>
> Without **−−if−exists**, attempting to delete a limit for zone that does not exist or doesn't have a limit is an error.  With **−−if−exists**, attempting to delete a limit that does not exist has no effect.

**list−zone−limits** *datapath*
> Prints the limits of all zones in *datapath*.

## Datapath Capabilities Command

The command query datapath capabilities.

**list−dp−cap** *datapath*
> Prints the datapath's capabilities.

## OpenFlow Controller Connectivity

**ovs−vswitchd** can perform all configured bridging and switching locally, or it can be configured to communicate with one or more external OpenFlow controllers.  The switch is typically configured to connect to a primary controller that takes charge of the bridge's flow table to implement a network policy.  In addition, the switch can be configured to listen to connections from service controllers.  Service controllers are typically used for occasional support and maintenance, e.g. with **ovs−ofctl**.

**get−controller** *bridge*
    Prints the configured controller target.

**del−controller** *bridge*
    Deletes the configured controller target.

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

**ssl:***host*[**:***port*]
**tcp:***host*[**:***port*]
    The specified *port* on the given *host*, which can be expressed either as a DNS name (if
    built with unbound library) or an IP address in IPv4 or IPv6 address format.  Wrap IPv6
    addresses in square brackets, e.g. **tcp:[::1]:6653**.  On Linux, use **%***device* to designate a
    scope for IPv6 link-level addresses, e.g. **tcp:[fe80::1234%eth0]:6653**.  For **ssl**, the
    **−−private−key**, **−−certificate**, and **−−ca−cert** options are mandatory.

    If *port* is not specified, it defaults to 6653.

**unix:***file*
    On POSIX, a Unix domain server socket named *file*.

    On Windows, connect to a local named pipe that is represented by a file created in the
    path *file* to mimic the behavior of a Unix domain socket.

**pssl:**[*port*][**:***host*]
**ptcp:**[*port*][**:***host*]
    Listens for OpenFlow connections on *port*.  The default *port* is 6653.  By default, connec-
    tions are allowed from any IPv4 address.  Specify *host* as an IPv4 address or a bracketed
    IPv6 address (e.g. **ptcp:6653:[::1]**).  On Linux, use **%***device* to designate a scope for
    IPv6 link-level addresses, e.g. **ptcp:6653:[fe80::1234%eth0]**.  DNS names can be used
    if built with unbound library.  For **pssl**, the **−−private−key**,**−−certificate**, and **−−ca−cert**
    options are mandatory.

**punix:***file*
    Listens for OpenFlow connections on the Unix domain server socket named *file*.

*Controller Failure Settings*

When a controller is configured, it is, ordinarily, responsible for setting up all flows on the switch.  Thus, if
the connection to the controller fails, no new network connections can be set up.  If the connection to the
controller stays down long enough, no packets can pass through the switch at all.

If the value is **standalone**, or if neither of these settings is set, **ovs−vswitchd** will take over responsibility
for setting up flows when no message has been received from the controller for three times the inactivity
probe interval.  In this mode, **ovs−vswitchd** causes the datapath to act like an ordinary MAC-learning
switch.  **ovs−vswitchd** will continue to retry connecting to the controller in the background and, when the
connection succeeds, it discontinues its standalone behavior.

If this option is set to **secure**, **ovs−vswitchd** will not set up flows on its own when the controller connection
fails.

**get−fail−mode** *bridge*
    Prints the configured failure mode.

**del−fail−mode** *bridge*
    Deletes the configured failure mode.

**set−fail−mode** *bridge* **standalone|secure**
    Sets the configured failure mode.

**Manager Connectivity**

These commands manipulate the **manager_options** column in the **Open_vSwitch** table and rows in the
**Managers** table.  When **ovsdb−server** is configured to use the **manager_options** column for OVSDB

connections (as described in the startup scripts provided with Open vSwitch; the corresponding **ovsdb−server** command option is **--remote=db:Open_vSwitch,Open_vSwitch,manager_options**), this allows the administrator to use **ovs−vsctl** 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 be an OVSDB active or passive connection method, e.g. **pssl:6640**, as described in **ovsdb**(7).

### SSL/TLS Configuration
When **ovs−vswitchd** is configured to connect over SSL/TLS for management or controller connectivity, the following parameters are required:

*private-key*
> Specifies a PEM file containing the private key used as the virtual switch's identity for SSL/TLS connections to the controller.

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

*ca-cert*  Specifies a PEM file containing the CA certificate used to verify that the virtual switch is connected to a trustworthy controller.

These files are read only once, at **ovs−vswitchd** startup time.  If their contents change, **ovs−vswitchd** must be killed and restarted.

These SSL/TLS settings apply to all SSL/TLS connections made by the virtual switch.

**get−ssl**  Prints the SSL/TLS configuration.

**del−ssl**  Deletes the current SSL/TLS configuration.

[**−−bootstrap**] **set−ssl** *private-key certificate ca-cert*
> Sets the SSL/TLS configuration.  The **−−bootstrap** option is described below.

*CA Certificate Bootstrap*

Ordinarily, all of the files named in the SSL/TLS configuration must exist when **ovs−vswitchd** starts.  However, if the *ca-cert* file does not exist and the **−−bootstrap** option is given, then **ovs−vswitchd** will attempt to obtain the CA certificate from the controller on its first SSL/TLS 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/TLS connections must be authenticated by a certificate signed by the CA certificate thus obtained.

**This option exposes the SSL/TLS 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 controller sends its CA certificate as part of the SSL/TLS certificate chain.  SSL/TLS protocols do not require the controller to send the CA certificate.

### Auto-Attach Commands
The IETF Auto-Attach SPBM draft standard describes a compact method of using IEEE 802.1AB Link Layer Discovery Protocol (LLDP) together with a IEEE 802.1aq Shortest Path Bridging (SPB) network to automatically attach network devices to individual services in a SPB network.  The intent here is to allow network applications and devices using OVS to be able to easily take advantage of features offered by industry standard SPB networks. A fundamental element of the Auto-Attach feature is to map traditional VLANs onto SPB I_SIDs. These commands manage the Auto-Attach I-SID/VLAN mappings.

**add−aa−mapping** *bridge i-sid vlan*
> Creates a new Auto-Attach mapping on *bridge* for *i-sid* and *vlan*.

**del−aa−mapping** *bridge i-sid vlan*
> Deletes an Auto-Attach mapping on *bridge* for *i-sid* and *vlan*.

**get−aa−mapping** *bridge*
> Lists all of the Auto-Attach mappings within *bridge* on standard output.

## 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 **ovs−vsctl** 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.

For a list of tables and their columns, see **ovs-vswitchd.conf.db**(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. **net** or **n** is sufficient to identify the **NetFlow** 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. For a column accepting a set of integers, database commands accept a range. A range is represented by two integers separated by **-**. A range is inclusive. A range has a maximum size of 4096 elements. If more elements are needed, they can be specified in separate ranges.

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* | *uuid*] **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 **ovs−vsctl** invocation in contexts where a UUID is expected.  Such references may

precede or follow the **create** command.

If a valid *uuid* is specified, then it is used as the UUID of the new row.

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.

## EXAMPLES
Create a new bridge named br0 and add port eth0 to it:

> **ovs−vsctl add−br br0**
> **ovs−vsctl add−port br0 eth0**

Alternatively, perform both operations in a single atomic transaction:

> **ovs−vsctl add−br br0 −− add−port br0 eth0**

Delete bridge **br0**, reporting an error if it does not exist:

> **ovs−vsctl del−br br0**

Delete bridge **br0** if it exists:

**ovs−vsctl −−if−exists del−br br0**

Set the **qos** column of the **Port** record for **eth0** to point to a new **QoS** record, which in turn points with its queue 0 to a new **Queue** record:

> **ovs−vsctl −− set port eth0 qos=@newqos −− −−id=@newqos create qos type=linux−htb other−config:max−rate=1000000 queues:0=@newqueue −− −−id=@newqueue create queue other−config:min−rate=1000000 other−config:max−rate=1000000**

## CONFIGURATION COOKBOOK

### Port Configuration

Add an ''internal port'' **vlan10** to bridge **br0** as a VLAN access port for VLAN 10, and configure it with an IP address:

> **ovs−vsctl add−port br0 vlan10 tag=10 −− set Interface vlan10 type=internal**

> **ip addr add 192.168.0.123/24 dev vlan10**

Add a GRE tunnel port **gre0** to remote IP address 1.2.3.4 to bridge **br0**:

> **ovs−vsctl add−port br0 gre0 −− set Interface gre0 type=gre options:remote_ip=1.2.3.4**

### Port Mirroring

Mirror all packets received or sent on **eth0** or **eth1** onto **eth2**, assuming that all of those ports exist on bridge **br0** (as a side-effect this causes any packets received on **eth2** to be ignored):

> **ovs−vsctl −− set Bridge br0 mirrors=@m \**

> **−− −−id=@eth0 get Port eth0 \**

> **−− −−id=@eth1 get Port eth1 \**

> **−− −−id=@eth2 get Port eth2 \**

> **−− −−id=@m create Mirror name=mymirror select-dst-port=@eth0,@eth1 select-src-port=@eth0,@eth1 output-port=@eth2**

Remove the mirror created above from **br0**, which also destroys the Mirror record (since it is now unreferenced):

> **ovs−vsctl −− −−id=@rec get Mirror mymirror \**

> **−− remove Bridge br0 mirrors @rec**

The following simpler command also works:

> **ovs−vsctl clear Bridge br0 mirrors**

### Quality of Service (QoS)

Create a **linux−htb** QoS record that points to a few queues and use it on **eth0** and **eth1**:

> **ovs−vsctl −− set Port eth0 qos=@newqos \**

> **−− set Port eth1 qos=@newqos \**

> **−− −−id=@newqos create QoS type=linux−htb other−config:max−rate=1000000000 queues=0=@q0,1=@q1 \**

> **−− −−id=@q0 create Queue other−config:min−rate=100000000 other−config:max−rate=100000000 \**

> **−− −−id=@q1 create Queue other−config:min−rate=500000000**

Deconfigure the QoS record above from **eth1** only:

> **ovs−vsctl clear Port eth1 qos**

To deconfigure the QoS record from both **eth0** and **eth1** and then delete the QoS record (which must be done explicitly because unreferenced QoS records are not automatically destroyed):

> **ovs−vsctl −− destroy QoS eth0 −− clear Port eth0 qos −− clear Port eth1 qos**

(This command will leave two unreferenced Queue records in the database.  To delete them, use "**ovs−vsctl list Queue**" to find their UUIDs, then "**ovs−vsctl destroy Queue** *uuid1 uuid2*" to destroy each of them or use "**ovs−vsctl -- --all destroy Queue**" to delete all records.)

**Connectivity Monitoring**

Monitor connectivity to a remote maintenance point on eth0.

> **ovs−vsctl set Interface eth0 cfm_mpid=1**

Deconfigure connectivity monitoring from above:

> **ovs−vsctl clear Interface eth0 cfm_mpid**

**NetFlow**

Configure bridge **br0** to send NetFlow records to UDP port 5566 on host 192.168.0.34, with an active time-out of 30 seconds:

> **ovs−vsctl −− set Bridge br0 netflow=@nf \**
>
> **−− −−id=@nf create NetFlow targets=\"192.168.0.34:5566\" active−timeout=30**

Update the NetFlow configuration created by the previous command to instead use an active timeout of 60 seconds:

> **ovs−vsctl set NetFlow br0 active_timeout=60**

Deconfigure the NetFlow settings from **br0**, which also destroys the NetFlow record (since it is now unref-erenced):

> **ovs−vsctl clear Bridge br0 netflow**

**sFlow**

Configure bridge **br0** to send sFlow records to a collector on 10.0.0.1 at port 6343, using **eth1**'s IP address as the source, with specific sampling parameters:

> **ovs−vsctl −− −−id=@s create sFlow agent=eth1 target=\"10.0.0.1:6343\" header=128 sampling=64 polling=10 \**
>
> **−− set Bridge br0 sflow=@s**

Deconfigure sFlow from **br0**, which also destroys the sFlow record (since it is now unreferenced):

> **ovs−vsctl −− clear Bridge br0 sflow**

**IPFIX**

Configure bridge **br0** to send one IPFIX flow record per packet sample to UDP port 4739 on host 192.168.0.34, with Observation Domain ID 123 and Observation Point ID 456, a flow cache active timeout of 1 minute (60 seconds), maximum flow cache size of 13 flows, and flows sampled on output port with tunnel info(sampling on input and output port is enabled by default if not disabled) :

> **ovs−vsctl −− set Bridge br0 ipfix=@i \**
>
> **−− −−id=@i create IPFIX targets=\"192.168.0.34:4739\" obs_domain_id=123 obs_point_id=456 cache_active_timeout=60 cache_max_flows=13 \**
>
> **other_config:enable-input-sampling=false other_config:enable-tunnel-sampling=true**

Deconfigure the IPFIX settings from **br0**, which also destroys the IPFIX record (since it is now unrefer-enced):

> **ovs−vsctl clear Bridge br0 ipfix**

**802.1D Spanning Tree Protocol (STP)**

Configure bridge **br0** to participate in an 802.1D spanning tree:

> **ovs−vsctl set Bridge br0 stp_enable=true**

Set the bridge priority of **br0** to 0x7800:

>       **ovs−vsctl set Bridge br0 other_config:stp-priority=0x7800**

Set the path cost of port **eth0** to 10:

>       **ovs−vsctl set Port eth0 other_config:stp-path-cost=10**

Deconfigure STP from above:

>       **ovs−vsctl set Bridge br0 stp_enable=false**

## Multicast Snooping

Configure bridge **br0** to enable multicast snooping:

>       **ovs−vsctl set Bridge br0 mcast_snooping_enable=true**

Set the multicast snooping aging time **br0** to 300 seconds:

>       **ovs−vsctl set Bridge br0 other_config:mcast-snooping-aging-time=300**

Set the multicast snooping table size **br0** to 2048 entries:

>       **ovs−vsctl set Bridge br0 other_config:mcast-snooping-table-size=2048**

Disable flooding of unregistered multicast packets to all ports. When set to **true**, the switch will send unregistered multicast packets only to ports connected to multicast routers. When it is set to **false**, the switch will send them to all ports. This command disables the flood of unregistered packets on bridge **br0**.

>       **ovs−vsctl set Bridge br0 other_config:mcast-snooping-disable-flood-unregistered=true**

Enable flooding of multicast packets (except Reports) on a specific port.

>       **ovs−vsctl set Port eth1 other_config:mcast-snooping-flood=true**

Enable flooding of Reports on a specific port.

>       **ovs−vsctl set Port eth1 other_config:mcast-snooping-flood-reports=true**

Deconfigure multicasting snooping from above:

>       **ovs−vsctl set Bridge br0 mcast_snooping_enable=false**

## 802.1D-2004 Rapid Spanning Tree Protocol (RSTP)

Configure bridge **br0** to participate in an 802.1D-2004 Rapid Spanning Tree:

>       **ovs−vsctl set Bridge br0 rstp_enable=true**

Set the bridge address of **br0** to 00:aa:aa:aa:aa:aa :

>       **ovs−vsctl set Bridge br0 other_config:rstp-address=00:aa:aa:aa:aa:aa**

Set the bridge priority of **br0** to 0x7000. The value must be specified in decimal notation and should be a multiple of 4096 (if not, it is rounded down to the nearest multiple of 4096). The default priority value is 0x800 (32768).

>       **ovs−vsctl set Bridge br0 other_config:rstp-priority=28672**

Set the bridge ageing time of **br0** to 1000 s. The ageing time value should be between 10 s and 1000000 s. The default value is 300 s.

>       **ovs−vsctl set Bridge br0 other_config:rstp-ageing-time=1000**

Set the bridge force protocol version of **br0** to 0. The force protocol version has two acceptable values: 0 (STP compatibility mode) and 2 (normal operation).

>       **ovs−vsctl set Bridge br0 other_config:rstp-force-protocol-version=0**

Set the bridge max age of **br0** to 10 s. The max age value should be between 6 s and 40 s. The default value is 20 s.

>       **ovs−vsctl set Bridge br0 other_config:rstp-max-age=10**

Set the bridge forward delay of **br0** to 15 s.  This value should be between 4 s and 30 s. The default value is 15 s.

>**ovs−vsctl set Bridge br0 other_config:rstp-forward-delay=15**

Set the bridge transmit hold count of **br0** to 7 s. This value should be between 1 s and 10 s. The default value is 6 s.

>**ovs−vsctl set Bridge br0 other_config:rstp-transmit-hold-count=7**

Enable RSTP on the Port **eth0**:

>**ovs−vsctl set Port eth0 other_config:rstp-enable=true**

Disable RSTP on the Port **eth0**:

>**ovs−vsctl set Port eth0 other_config:rstp-enable=false**

Set the priority of port **eth0** to 32. The value must be specified in decimal notation and should be a multiple of 16 (if not, it is rounded down to the nearest multiple of 16). The default priority value is 0x80 (128).

>**ovs−vsctl set Port eth0 other_config:rstp-port-priority=32**

Set the port number of port **eth0** to 3:

>**ovs−vsctl set Port eth0 other_config:rstp-port-num=3**

Set the path cost of port **eth0** to 150:

>**ovs−vsctl set Port eth0 other_config:rstp-path-cost=150**

Set the admin edge value of port **eth0**:

>**ovs−vsctl set Port eth0 other_config:rstp-port-admin-edge=true**

Set the auto edge value of port **eth0**:

>**ovs−vsctl set Port eth0 other_config:rstp-port-auto-edge=true**

Set the admin point to point MAC value of port **eth0**. Acceptable values are **0** (not point-to-point), **1** (point-to-point, the default value) or **2** (automatic detection). The auto-detection mode is not currently implemented, and the value **2** has the same effect of **0** (not point-to-point).

>**ovs−vsctl set Port eth0 other_config:rstp-admin-p2p-mac=1**

Set the admin port state value of port **eth0**. **true** is the default value.

>**ovs−vsctl set Port eth0 other_config:rstp-admin-port-state=false**

Set the mcheck value of port **eth0**:

>**ovs−vsctl set Port eth0 other_config:rstp-port-mcheck=true**

Deconfigure RSTP from above:

>**ovs−vsctl set Bridge br0 rstp_enable=false**

**OpenFlow Version**
>Configure bridge **br0** to support OpenFlow versions 1.0, 1.2, and 1.3:

>>**ovs−vsctl set bridge br0 protocols=OpenFlow10,OpenFlow12,OpenFlow13**

**Flow Table Configuration**
>Make flow table 0 on bridge br0 refuse to accept more than 100 flows:

>>**ovs−vsctl −− −−id=@ft create Flow_Table flow_limit=100 overflow_policy=refuse −− set Bridge br0 flow_tables=0=@ft**

>Make flow table 0 on bridge br0 evict flows, with fairness based on the matched ingress port, when there are more than 100:

>>**ovs−vsctl −− −−id=@ft create Flow_Table flow_limit=100 overflow_policy=evict groups='"NXM_OF_IN_PORT[]"' −− set Bridge br0 flow_tables:0=@ft**

**EXIT STATUS**

0            Successful program execution.

1            Usage, syntax, or configuration file error.

2            The *bridge* argument to **br−exists** specified the name of a bridge that does not exist.

65 (Linux and BSD) or 160 (Windows)
             The database transaction was committed successfully, but **ovs−vsctl** failed while waiting for
             **ovs−vswitchd** to finish reconfiguring itself to reflect the changes.  This typically happens when
             **--no−wait** is not specified and the daemon does not acknowledge the update in time or encounters
             an error applying it.

             Note that the change remains recorded in the database and is not rolled back.  This exit code is in-
             tended to signal that while the database was updated, the system configuration may not yet reflect
             the intended changes.

**SEE ALSO**

             **ovsdb−server**(1), **ovs−vswitchd**(8), **ovs−vswitchd.conf.db**(5).