NAME

ovsdb−client − command-line interface to **ovsdb-server**(1)

SYNOPSIS

Server-Level Commands:

ovsdb−client [*options*] **list−dbs** [*server*]

Database Schema Commands:

ovsdb−client [*options*] **get−schema** [*server*] [*database*]
ovsdb−client [*options*] **list−tables** [*server*] [*database*]
ovsdb−client [*options*] **list−columns** [*server*] [*database*] [*table*]

Database Version Management Commands:

ovsdb−client [*options*] **convert** [*server*] *schema*
ovsdb−client [*options*] **needs−conversion** [*server*] *schema*
ovsdb−client [*options*] **get−schema−version** [*server*] [*database*]

Data Management Commands:

ovsdb−client [*options*] **transact** [*server*] *transaction*
ovsdb−client [*options*] **query** [*server*] *transaction*
ovsdb−client [*options*] **dump** [*server*] [*database*] [*table* [*column*...]]
ovsdb−client [*options*] **backup** [*server*] [*database*] **>** *snapshot*
ovsdb−client [*options*] [**−−force**] **restore** [*server*] [*database*] **<** *snapshot*
ovsdb−client [*options*] **monitor** [*server*] [*database*] *table* [*column*[**,***column*]...]...
ovsdb−client [*options*] **monitor** [*server*] [*database*] **ALL**
ovsdb−client [*options*] **monitor−cond** [*server*] [*database*] *conditions table* [*column*[**,***column*]...]...
ovsdb−client [*options*] **monitor−cond−since** [*server*] [*database*] [*last-id*] *conditions table* [*column*[**,***column*]...]...
ovsdb−client [*options*] **wait** [*server*] *database state*

Testing Commands:

ovsdb−client [*options*] **lock** [*server*] *lock*
ovsdb−client [*options*] **steal** [*server*] *lock*
ovsdb−client [*options*] **unlock** [*server*] *lock*

Other Commands:

ovsdb−client help

Cluster Options:

[**−−no−leader−only**]

Output formatting options:

[**−−format=***format*] [**−−data=***format*] [**−−no-headings**] [**−−pretty**] [**−−bare**] [**−−timestamp**]

Daemon options:

[**−−pidfile**[**=***pidfile*]] [**−−overwrite−pidfile**] [**−−detach**] [**−−no−chdir**] [**−−no−self−confinement**]

Logging options:

[**−v**[*module*[**:***destination*[**:***level*]]]]...
[**−−verbose[=***module*[**:***destination*[**:***level*]]]]...
[**−−log−file**[**=***file*]]

Public key infrastructure options:

[**−−private−key=***privkey.pem*]
[**−−certificate=***cert.pem*]
[**−−ca−cert=***cacert.pem*]
[**−−bootstrap−ca−cert=***cacert.pem*]

SSL/TLS connection options:

[**−−ssl−protocols=***protocols*]
[**−−ssl−ciphers=***ciphers*]
[**−−ssl−ciphersuites=***ciphersuites*]

Replay options:
[**−−record**[**=***directory*]] [**−−replay**[**=***directory*]]

Common options:
[**−h** | **−−help**] [**−V** | **−−version**]

DESCRIPTION

The **ovsdb−client** program is a command-line client for interacting with a running **ovsdb−server** process. Each command connects to the specified OVSDB *server*, which may be an OVSDB active or passive connection method, as described in **ovsdb**(7). The default server is **unix:/usr/local/var/run/openvswitch/db.sock** and the default *database* is **Open_vSwitch**.

ovsdb−client supports the *method1***,***method2***,...,***methodN* syntax described in **ovsdb**(7) for connecting to a cluster. When this syntax is used, **ovsdb−client** tries the cluster members in random order until it finds the cluster leader. Specify the **−−no−leader−only** option to instead accept any server that is connected to the cluster.

For an introduction to OVSDB and its implementation in Open vSwitch, see **ovsdb**(7).

The following sections describe the commands that **ovsdb−client** supports.

Server-Level Commands

Most **ovsdb−client** commands work with an individual database, but these commands apply to an entire database server.

list−dbs [*server*]
Connects to *server*, retrieves the list of known databases, and prints them one per line. These database names are the ones that other commands may use for *database*.

Database Schema Commands

These commands obtain the schema from a database and print it or part of it.

get−schema [*server*] [*database*]
Connects to *server*, retrieves the schema for *database*, and prints it in JSON format.

list−tables [*server*] [*database*]
Connects to *server*, retrieves the schema for *database*, and prints a table listing the name of each table within the database.

list−columns [*server*] [*database*] *table*
Connects to *server*, retrieves the schema for *database*, and prints a table listing the name and type of each column. If *table* is specified, only columns in that table are listed; otherwise, the tables include columns in all tables.

Database Version Management Commands

An OVSDB schema has a schema version number, and an OVSDB database embeds a particular version of an OVSDB schema. These version numbers take the form *x***.***y***.***z*, e.g. **1.2.3**. The OVSDB implementation does not enforce a particular version numbering scheme, but schemas managed within the Open vSwitch project use the following approach. Whenever the database schema is changed in a non-backward compatible way (e.g. deleting a column or a table), *x* is incremented (and *y* and *z* are reset to 0). When the database schema is changed in a backward compatible way (e.g. adding a new column), *y* is incremented (and *z* is reset to 0). When the database schema is changed cosmetically (e.g. reindenting its syntax), *z* is incremented.

Some OVSDB databases and schemas, especially very old ones, do not have a version number.

Schema version numbers and Open vSwitch version numbers are independent.

These commands work with different versions of OVSDB schemas and databases.

convert [*server*] *schema*
Reads an OVSDB schema in JSON format, as specified in the OVSDB specification, from *schema*, then connects to *server* and requests the server to convert the database whose name is specified in *schema* to the schema also specified in *schema*.

The conversion is atomic, consistent, isolated, and durable. Following the schema change, the server notifies clients that use the **set_db_change_aware** RPC introduced in Open vSwitch 2.9 and cancels their outstanding transactions and monitors. The server disconnects other clients, enabling them to notice the change when they reconnect.

This command can do simple ‘‘upgrades’’ and ‘‘downgrades’’ on a database’s schema. The data in the database must be valid when interpreted under *schema*, with only one exception: data for tables and columns that do not exist in *schema* are ignored. Columns that exist in *schema* but not in the database are set to their default values. All of *schema*’s constraints apply in full.

Some uses of this command can cause unrecoverable data loss. For example, converting a database from a schema that has a given column or table to one that does not will delete all data in that column or table. Back up critical databases before converting them.

This command works with clustered and standalone databases. Standalone databases may also be converted (offline) with **ovsdb−tool**’s **convert** command.

needs−conversion [*server*] *schema*
Reads the schema from *schema*, then connects to *server* and requests the schema from the database whose name is specified in *schema*. If the two schemas are the same, prints **no** on stdout; if they differ, prints **yes**.

get−schema−version [*server*] [*database*]
Connects to *server*, retrieves the schema for *database*, and prints its version number on stdout. If *database* was created before schema versioning was introduced, then it will not have a version number and this command will print a blank line.

get−schema−cksum [*server*] [*database*]
Connects to *server*, retrieves the schema for *database*, and prints its checksum on stdout. If *database* does not include a checksum, prints a blank line.

Data Management Commands
These commands read or modify the data in a database.

transact [*server*] *transaction*
Connects to *server*, sends it the specified *transaction*, which must be a JSON array appropriate for use as the **params** to a JSON-RPC **transact** request, and prints the received reply on stdout.

query [*server*] *transaction*
This commands acts like a read-only version of **transact**. It connects to *server*, sends it the specified *transaction*, which must be a JSON array appropriate for use as the **params** to a JSON-RPC **transact** request, and prints the received reply on stdout. To ensure that the transaction does not modify the database, this command appends an **abort** operation to the set of operations included in *transaction* before sending it to the database, and then removes the **abort** result from the reply (if it is present).

dump [*server*] [*database*] [*table* [*column*...]]
Connects to *server*, retrieves all of the data in *database*, and prints it on stdout as a series of tables. If *table* is specified, only that table is retrieved. If at least one *column* is specified, only those columns are retrieved.

backup [*server*] [*database*] **>** *snapshot*
Connects to *server*, retrieves a snapshot of the schema and data in *database*, and prints it on stdout in the format used for OVSDB standalone and active-backup databases. This is an appropriate way to back up any remote database. The database snapshot that it outputs is suitable to be served up directly by **ovsdb−server** or used as the input to **ovsdb−client restore**.

Another way to back up a standalone or active-backup database is to copy its database file, e.g. with **cp**. This is safe even if the database is in use.

The output does not include ephemeral columns, which by design do not survive across restarts of **ovsdb−server**.

[**−−force**] **restore** [*server*] [*database*] < *snapshot*

Reads *snapshot*, which must be a OVSDB standalone or active-backup database (possibly but not necessarily created by **ovsdb−client backup). Then, connects to** *server*, verifies that *database* and *snapshot* have the same schema, then deletes all of the data in *database* and replaces it by *snapshot*. The replacement happens atomically, in a single transaction.

UUIDs for rows in the restored database will differ from those in *snapshot*, because the OVSDB protocol does not allow clients to specify row UUIDs. Another way to restore a standalone or active-backup database, which does also restore row UUIDs, is to stop the server or servers, replace the database file by the snapshot, then restart the database. Either way, ephemeral columns are not restored, since by design they do not survive across restarts of **ovsdb−server**.

Normally **restore** exits with a failure if **snapshot** and the server's database have different schemas. In such a case, it is a good idea to convert the database to the new schema before restoring, e.g. with **ovsdb−client convert**. Use **−−force** to proceed regardless of schema differences even though the restore might fail with an error or succeed with surprising results.

monitor [*server*] [*database*] *table* [*column*[**,***column*]...]...
monitor−cond [*server*] [*database*] *conditions table* [*column*[**,***column*]...]...
monitor−cond−since [*server*] [*database*] [*last-id*] *conditions table* [*column*[**,***column*]...]...

Connects to *server* and monitors the contents of rows that match conditions in *table* in *database*. By default, the initial contents of *table* are printed, followed by each change as it occurs. If conditions empty, all rows will be monitored. If at least one *column* is specified, only those columns are monitored. The following *column* names have special meanings:

!initial Do not print the initial contents of the specified columns.

!insert Do not print newly inserted rows.

!delete Do not print deleted rows.

!modify
Do not print modifications to existing rows.

Multiple [*column*[**,***column*]...] groups may be specified as separate arguments, e.g. to apply different reporting parameters to each group. Whether multiple groups or only a single group is specified, any given column may only be mentioned once on the command line.

conditions is a JSON array of <condition> as defined in RFC 7047 5.1 with the following change: A condition can be either a 3-element JSON array as described in the RFC or a boolean value.

If **−−detach** is used with **monitor**, **monitor−cond** or **monitor−cond−since**, then **ovsdb−client** detaches after it has successfully received and printed the initial contents of *table*.

The **monitor** command uses RFC 7047 "monitor" method to open a monitor session with the server. The **monitor−cond** and **monitor−cond−since** commandls uses RFC 7047 extension "monitor_cond" and "monitor_cond_since" methods. See **ovsdb−server**(1) for details.

monitor [*server*] [*database*] **ALL**

Connects to *server* and monitors the contents of all tables in *database*. Prints initial values and all kinds of changes to all columns in the database. The **−−detach** option causes **ovsdb−client** to detach after it successfully receives and prints the initial database contents.

The **monitor** command uses RFC 7047 "monitor" method to open a monitor session with the server.

wait [*server*] *database state*

Waits for *database* on *server* to enter a desired *state*, which may be one of:

added Waits until a database with the given name has been added to *server*.

connected
Waits until a database with the given name has been added to *server*. Then, if *database* is clustered, additionally waits until it has joined and connected to its cluster.

removed

Waits until *database* has been removed from the database server. This can also be used to wait for a database to complete leaving its cluster, because **ovsdb−server** removes a database at that point.

database is mandatory for this command because it is often used to check for databases that have not yet been added to the server, so that the **ovsdb−client** semantics of acting on a default database do not work.

This command acts on a particular database server, not on a cluster, so *server* must name a single server, not a comma-delimited list of servers.

Testing commands

These commands are mostly of interest for testing the correctness of the OVSDB server.

lock [*server*] *lock*
steal [*server*] *lock*
unlock [*server*] *lock*

Connects to *server* and issues corresponding RFC 7047 lock operations on *lock*. Prints json reply or subsequent update messages. The **−−detach** option causes **ovsdb−client** to detach after it successfully receives and prints the initial reply.

When running with the **−−detach** option, **lock**, **steal**, **unlock** and **exit** commands can be issued by using **ovs-appctl**. **exit** command causes the **ovsdb−client** to close its **ovsdb−server** connection before exit. The **lock**, **steal** and **unlock** commands can be used to issue additional lock operations over the same **ovsdb−server** connection. All above commands take a single *lock* argument, which does not have to be the same as the *lock* that **ovsdb−client** started with.

OPTIONS

Output Formatting Options

Much of the output from **ovsdb−client** is in the form of tables. The following options controlling output formatting:

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

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

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

json The RFC 4627 JSON format as described above.

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

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

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

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

−−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.

−−timestamp
For the **monitor**, **monitor−cond** and **monitor−cond−since** commands, add a timestamp to each table update. Most output formats add the timestamp on a line of its own just above the table. The JSON output format puts the timestamp in a member of the top-level JSON object named **time**.

−t
−−timeout=*secs*
Limits **ovsdb−client** runtime to approximately *secs* seconds. If the timeout expires, **ovsdb−client** will exit with a **SIGALRM** signal.

Daemon Options
The daemon options apply only to the **monitor**, **monitor−cond** and **monitor−cond−since** commands. With any other command, they have no effect.

The following options are valid on POSIX based platforms.

−−pidfile[**=***pidfile*]
Causes a file (by default, **ovsdb−client.pid**) to be created indicating the PID of the running process. If the *pidfile* argument is not specified, or if it does not begin with **/**, then it is created in **/usr/local/var/run/openvswitch**.

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

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

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

−−detach
Runs **ovsdb−client** as a background process. The process forks, and in the child it starts a new session, closes the standard file descriptors (which has the side effect of disabling logging to the

console), and changes its current directory to the root (unless **−−no−chdir** is specified). After the child completes its initialization, the parent exits.

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

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

−−no−chdir
By default, when **−−detach** is specified, **ovsdb−client** changes its current working directory to the root directory after it detaches. Otherwise, invoking **ovsdb−client** from a carelessly chosen directory would prevent the administrator from unmounting the file system that holds that directory.

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

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

−−no−self−confinement
By default daemon will try to self-confine itself to work with files under well-known directories determined during build. It is better to stick with this default behavior and not to use this flag unless some other Access Control is used to confine daemon. Note that in contrast to other access control implementations that are typically enforced from kernel-space (e.g. DAC or MAC), self-confinement is imposed from the user-space daemon itself and hence should not be considered as a full confinement strategy, but instead should be viewed as an additional layer of security.

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

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

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

Logging Options

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

- A valid module name, as displayed by the **vlog/list** command on **ovs−appctl**(8), limits the log level change to the specified module.
- **syslog**, **console**, or **file**, to limit the log level change to only to the system log, to the console, or to a file, respectively. (If **−−detach** is specified, **ovsdb−client** 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/ovsdb−client.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**.

Public Key Infrastructure Options
−p *privkey.pem*
−−private−key=*privkey.pem*
Specifies a PEM file containing the private key used as **ovsdb−client**'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 **ovsdb−client** 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 **ovsdb−client** 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**.

SSL/TLS Connection Options
−−ssl−protocols=*protocols*
Specifies a range or a comma- or space-delimited list of the SSL/TLS protocols **ovsdb−client** will enable for SSL/TLS connections. Supported *protocols* include **TLSv1.2** and **TLSv1.3**. Ranges can be provided in a form of two protocol names separated with a dash, or as a single protocol name with a plus sign. For example, use **TLSv1.2-TLSv1.3** to allow **TLSv1.2** and **TLSv1.3**. Use **TLSv1.2+** to allow **TLSv1.2** and any later protocol. The option accepts a list of protocols or exactly one range. The range is a preferred way of specifying protocols and the option always behaves as if the range between the minimum and the maximum specified version is provided, i.e., if the option is set to **TLSv1.X,TLSv1.(X+2)**, the **TLSv1.(X+1)** will also be enabled as if it was a range. Regardless of order, the highest protocol supported by both sides will be chosen when making the connection. The default when this option is omitted is **TLSv1.2** or later.

−−ssl−ciphers=*ciphers*
Specifies, in OpenSSL cipher string format, the ciphers **ovsdb−client** will support for SSL/TLS connections with TLSv1.2. The default when this option is omitted is **DEFAULT:@SECLEVEL=2**.

−−ssl−ciphersuites=*ciphersuites*
Specifies, in OpenSSL ciphersuite string format, the ciphersuites **ovsdb−client** will support for SSL/TLS connections with TLSv1.3 and later. Default value from OpenSSL will be used when this option is omitted.

Other Options
−−record[=*directory*]
Sets the process in "recording" mode, in which it will record all the connections, data from streams (Unix domain and network sockets) and some other important necessary bits, so they could be

replayed later. Recorded data is stored in replay files in specified *directory*. If *directory* does not begin with **/**, it is interpreted as relative to **/usr/local/var/run/openvswitch**. If *directory* is not specified, **/usr/local/var/run/openvswitch** will be used.

−−replay[=*directory*]
Sets the process in "replay" mode, in which it will read information about connections, data from streams (Unix domain and network sockets) and some other necessary bits directly from replay files instead of using real sockets. Replay files from the *directory* will be used. If *directory* does not begin with **/**, it is interpreted as relative to **/usr/local/var/run/openvswitch**. If *directory* is not specified, **/usr/local/var/run/openvswitch** will be used.

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

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

SEE ALSO

ovsdb(7), **ovsdb−server**(1), **ovsdb−client**(1).