NAME
ovs-ofctl − administer OpenFlow switches

SYNOPSIS
ovs-ofctl [options] command [switch] [args...]

DESCRIPTION
The ovs-ofctl program is a command line tool for monitoring and administering OpenFlow switches. It can also show the current state of an OpenFlow switch, including features, configuration, and table entries. It should work with any OpenFlow switch, not just Open vSwitch.

OpenFlow Switch Management Commands
These commands allow ovs-ofctl to monitor and administer an OpenFlow switch. It is able to show the current state of a switch, including features, configuration, and table entries.

Most of these commands take an argument that specifies the method for connecting to an OpenFlow switch. The following connection methods are supported:

- **ssl:** 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.

- **tcp:** host[:port]
  - 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.

- **unix:** file
  - This is short for `unix::file`, as long as file does not contain a colon.

- **bridge**
  - This is short for `unix::usr/local/var/run/openvswitch/bridge.mgmt`, as long as bridge does not contain a colon.

- **[type@]dp**
  - Attempts to look up the bridge associated with dp and open as above. If type is given, it specifies the datapath provider of dp, otherwise the default provider system is assumed.

- **show switch**
  - Prints to the console information on switch, including information on its flow tables and ports.

- **dump−tables switch**
  - Prints to the console statistics for each of the flow tables used by switch.

- **dump−table−features switch**
  - Prints to the console features for each of the flow tables used by switch.

- **dump−table−desc switch**
  - Prints to the console configuration for each of the flow tables used by switch for OpenFlow 1.4+.

- **mod−table switch table setting**
  - This command configures flow table settings in switch for OpenFlow table table, which may be expressed as a number or (unless `−−no−names` is specified) a name.
  - The available settings depend on the OpenFlow version in use. In OpenFlow 1.1 and 1.2 (which must be enabled with the `−O` option) only, mod−table configures behavior when no flow is found when a packet is looked up in a flow table. The following setting values are available:

  - **drop** Drop the packet.
continue
Continue to the next table in the pipeline. (This is how an OpenFlow 1.0 switch always handles packets that do not match any flow, in tables other than the last one.)

controller
Send to controller. (This is how an OpenFlow 1.0 switch always handles packets that do not match any flow in the last table.)

In OpenFlow 1.3 and later (which must be enabled with the −O option) and Open vSwitch 2.11 and later only, mod−table can change the name of a table:

name: new-name
Changes the name of the table to new-name. Use an empty new-name to clear the name. (This will be ineffective if the name is set via the name column in the Flow_Table table in the Open_vSwitch database as described in ovs-vswitchd.conf.db(5).)

In OpenFlow 1.4 and later (which must be enabled with the −O option) only, mod−table configures the behavior when a controller attempts to add a flow to a flow table that is full. The following setting values are available:

evict Delete some existing flow from the flow table, according to the algorithm described for the Flow_Table table in ovs-vswitchd.conf.db(5).

noevict Refuse to add the new flow. (Eviction might still be enabled through the overflow_policy column in the Flow_Table table documented in ovs-vswitchd.conf.db(5).)

vacancy: low, high
Enables sending vacancy events to controllers using TABLE_STATUS messages, based on percentage thresholds low and high.

novacancy
Disables vacancy events.

dump−ports switch [netdev]
Prints to the console statistics for network devices associated with switch. If netdev is specified, only the statistics associated with that device will be printed. netdev can be an OpenFlow assigned port number or device name, e.g. eth0.

dump−ports−desc switch [port]
Prints to the console detailed information about network devices associated with switch. To dump only a specific port, specify its number as port. Otherwise, if port is omitted, or if it is specified as ANY, then all ports are printed. This is a subset of the information provided by the show command.

If the connection to switch negotiates OpenFlow 1.0, 1.2, or 1.2, this command uses an OpenFlow extension only implemented in Open vSwitch (version 1.7 and later).

Only OpenFlow 1.5 and later support dumping a specific port. Earlier versions of OpenFlow always dump all ports.

mod−port switch port action
Modify characteristics of port port in switch. port may be an OpenFlow port number or name (unless −−no−names is specified) or the keyword LOCAL (the preferred way to refer to the OpenFlow local port). The action may be any one of the following:

up
down Enable or disable the interface. This is equivalent to ip link set up or ip link set down on a Unix system.

stp
no−stp Enable or disable 802.1D spanning tree protocol (STP) on the interface. OpenFlow implementations that don’t support STP will refuse to enable it.
receive
no−receive
receive−stp
no−receive−stp
Enable or disable OpenFlow processing of packets received on this interface. When packet processing is disabled, packets will be dropped instead of being processed through the OpenFlow table. The receive or no−receive setting applies to all packets except 802.1D spanning tree packets, which are separately controlled by receive−stp or no−receive−stp.

forward
no−forward
Allow or disallow forwarding of traffic to this interface. By default, forwarding is enabled.

flood
no−flood
Controls whether an OpenFlow flood action will send traffic out this interface. By default, flooding is enabled. Disabling flooding is primarily useful to prevent loops when a spanning tree protocol is not in use.

packet−in
no−packet−in
Controls whether packets received on this interface that do not match a flow table entry generate a “packet in” message to the OpenFlow controller. By default, “packet in” messages are enabled.

The show command displays (among other information) the configuration that mod−port changes.

get−frags switch
Prints switch’s fragment handling mode. See set−frags, below, for a description of each fragment handling mode.

The show command also prints the fragment handling mode among its other output.

set−frags switch frag_mode
Configures switch’s treatment of IPv4 and IPv6 fragments. The choices for frag_mode are:

normal
Fragments pass through the flow table like non-fragmented packets. The TCP ports, UDP ports, and ICMP type and code fields are always set to 0, even for fragments where that information would otherwise be available (fragments with offset 0). This is the default fragment handling mode for an OpenFlow switch.

drop
Fragments are dropped without passing through the flow table.

reassemble
The switch reassembles fragments into full IP packets before passing them through the flow table. Open vSwitch does not implement this fragment handling mode.

nx−match
Fragments pass through the flow table like non-fragmented packets. The TCP ports, UDP ports, and ICMP type and code fields are available for matching for fragments with offset 0, and set to 0 in fragments with nonzero offset. This mode is a Nicira extension.

See the description of ip_frag, in ovs−fields(7), for a way to match on whether a packet is a fragment and on its fragment offset.

dump−flows switch [flows]
Prints to the console all flow entries in switch’s tables that match flows. If flows is omitted, all flows in the switch are retrieved. See Flow Syntax, below, for the syntax of flows. The output
format is described in Table Entry Output.

By default, `ovs-ofctl` prints flow entries in the same order that the switch sends them, which is unlikely to be intuitive or consistent. Use `--sort` and `--rsort` to control display order. The `--names/--no-names` and `--stats/--no-stats` options also affect output formatting. See the descriptions of these options, under OPTIONS below, for more information.

**dump-aggregate switch [flows]**
Prints to the console aggregate statistics for flows in switch’s tables that match flows. If flows is omitted, the statistics are aggregated across all flows in the switch’s flow tables. See Flow Syntax, below, for the syntax of flows. The output format is described in Table Entry Output.

**queue-stats switch [port [queue]]**
Prints to the console the statistics for the specified queue on port within switch. port can be an OpenFlow port number or name, the keyword LOCAL (the preferred way to refer to the OpenFlow local port), or the keyword ALL. Either of port or queue or both may be omitted (or equivalently the keyword ALL). If both are omitted, statistics are printed for all queues on all ports. If only queue is omitted, then statistics are printed for all queues on port; if only port is omitted, then statistics are printed for queue on every port where it exists.

**queue-get-config switch [port [queue]]**
Prints to the console the configuration of queue on port in switch. If port is omitted or ANY, reports queues for all port. If queue is omitted or ANY, reports all queues. For OpenFlow 1.3 and earlier, the output always includes all queues, ignoring queue if specified.

This command has limited usefulness, because ports often have no configured queues and because the OpenFlow protocol provides only very limited information about the configuration of a queue.

**dump-ipfix-bridge switch**
Prints to the console the statistics of bridge IPFIX for switch. If bridge IPFIX is configured on the switch, IPFIX statistics can be retrieved. Otherwise, error message will be printed.

This command uses an Open vSwitch extension that is only in Open vSwitch 2.6 and later.

**dump-ipfix-flow switch**
Prints to the console the statistics of flow-based IPFIX for switch. If flow-based IPFIX is configured on the switch, statistics of all the collector set ids on the switch will be printed. Otherwise, print error message.

Refer to ovs-vswitchd.conf.db(5) for more details on configuring flow based IPFIX and collector set ids.

This command uses an Open vSwitch extension that is only in Open vSwitch 2.6 and later.

**ct-flush-zone switch zone**
Flushes the connection tracking entries in zone on switch.

This command uses an Open vSwitch extension that is only in Open vSwitch 2.6 and later.

**OpenFlow Switch Flow Table Commands**
These commands manage the flow table in an OpenFlow switch. In each case, flow specifies a flow entry in the format described in Flow Syntax, below; file is a text file that contains zero or more flows in the same syntax, one per line, and the optional `--bundle` option operates the command as a single atomic translation, see option `--bundle`, below.

`[--bundle] add-flow switch flow`
`[--bundle] add-flow switch –< file`
`[--bundle] add-flows switch file`
Add each flow entry to switch’s tables. Each flow specification (e.g., each line in file) may start with `add`, `modify`, `delete`, `modify_strict`, or `delete_strict` keyword to specify whether a flow is to be added, modified, or deleted, and whether the modify or delete is strict or not. For backwards compatibility a flow specification without one of these keywords is treated as a flow add. All flow mods are executed in the order specified.
ovs-ofctl(8) Open vSwitch Manual ovs-ofctl(8)

[---bundle] [---strict] mod−flows switch flow

Modify the actions in entries from switch’s tables that match the specified flows. With ---strict, wildcards are not treated as active for matching purposes.

[---bundle] del−flows switch

Deletes entries from switch’s flow table. With only a switch argument, deletes all flows. Otherwise, deletes flow entries that match the specified flows. With ---strict, wildcards are not treated as active for matching purposes.

[---bundle] [---readd] replace−flows switch file

Reads flow entries from file (or stdin if file is −) and queries the flow table from switch. Then it fixes up any differences, adding flows from flow that are missing on switch, deleting flows from switch that are not in file, and updating flows in switch whose actions, cookie, or timeouts differ in file.

With ---readd, ovs-ofctl adds all the flows from file, even those that exist with the same actions, cookie, and timeout in switch. In OpenFlow 1.0 and 1.1, re-adding a flow always resets the flow’s packet and byte counters to 0, and in OpenFlow 1.2 and later, it does so only if the reset_counts flag is set.

diff−flows source1 source2

Reads flow entries from source1 and source2 and prints the differences. A flow that is in source1 but not in source2 is printed preceded by a −, and a flow that is in source2 but not in source1 is printed preceded by a +. If a flow exists in both source1 and source2 with different actions, cookie, or timeouts, then both versions are printed preceded by − and +, respectively.

source1 and source2 may each name a file or a switch. If a name begins with / or ., then it is considered to be a file name. A name that contains : is considered to be a switch. Otherwise, it is a file if a file by that name exists, a switch if not.

For this command, an exit status of 0 means that no differences were found, 1 means that an error occurred, and 2 means that some differences were found.

packet−out switch packet−out

Connects to switch and instructs it to execute the packet-out OpenFlow message, specified as defined in Packet−Out Syntax section.

Group Table Commands

These commands manage the group table in an OpenFlow switch. In each case, group specifies a group entry in the format described in Group Syntax, below, and file is a text file that contains zero or more groups in the same syntax, one per line, and the optional ---bundle option operates the command as a single atomic transaction, see option ---bundle, below.

The group commands work only with switches that support OpenFlow 1.1 or later or the Open vSwitch group extensions to OpenFlow 1.0 (added in Open vSwitch 2.9.90). For OpenFlow 1.1 or later, it is necessary to explicitly enable these protocol versions in ovs-ofctl (using −O). For more information, see “Q: What versions of OpenFlow does Open vSwitch support?” in the Open vSwitch FAQ.

[---bundle] add−group switch group

Add each group entry to switch’s tables. Each group specification (e.g., each line in file) may start with add, modify, add_or_mod, delete, insert_bucket, or remove_bucket keyword to specify whether a flow is to be added, modified, or deleted, or whether a group bucket is to be added or removed. For backwards compatibility a group specification without one of these keywords is treated as a group add. All group mods are executed in the order specified.
Modify the action buckets in entries from `switch`’s tables for each group entry. If a specified group does not already exist, then without `--may-create`, this command has no effect; with `--may-create`, it creates a new group. The `--may-create` option uses an Open vSwitch extension to OpenFlow only implemented in Open vSwitch 2.6 and later.

Deletes entries from `switch`’s group table. With only a `switch` argument, deletes all groups. Otherwise, deletes the group for each group entry.

Add buckets to an existing group present in the `switch`’s group table. If no `command_bucket_id` is present in the group specification then all buckets of the group are removed.

Remove buckets to an existing group present in the `switch`’s group table. If no `command_bucket_id` is present in the group specification then all buckets of the group are removed.

Print group entries in `switch`’s tables to console. To dump only a specific group, specify its number as `group`. Otherwise, if `group` is omitted, or if it is specified as `ALL`, then all groups are printed.

Prints to the console the group features of the `switch`.

Prints to the console statistics for the specified `group` in `switch`’s tables. If `group` is omitted then statistics for all groups are printed.

These commands manage the meter table in an OpenFlow switch. In each case, `meter` specifies a meter entry in the format described in Meter Syntax, below.

OpenFlow 1.3 introduced support for meters, so these commands only work with switches that support OpenFlow 1.3 or later. It is necessary to explicitly enable these protocol versions in `ovs-ofctl` (using `-O`) and in the switch itself (with the `protocols` column in the Bridge table). For more information, see “Q: What versions of OpenFlow does Open vSwitch support?” in the Open vSwitch FAQ.

Add a meter entry to `switch`’s tables. The `meter` syntax is described in section Meter Syntax, below.

Modify an existing meter.

Delete entries from `switch`’s meter table. To delete only a specific meter, specify its number as `meter`. Otherwise, if `meter` is omitted, or if it is specified as `all`, then all meters are deleted.

Print entries from `switch`’s meter table. To print only a specific meter, specify its number as `meter`. Otherwise, if `meter` is omitted, or if it is specified as `all`, then all meters are printed.
meter-stats switch [meter]
Print meter statistics. meter can specify a single meter with syntax meter=id, or all meters with syntax meter=all.

meter-features switch
Print meter features.

OpenFlow Switch Bundle Command
Transactional updates to both flow and group tables can be made with the bundle command. file is a text file that contains zero or more flow mods, group mods, or packet-outs in Flow Syntax, Group Syntax, or Packet-Out Syntax, each line preceded by flow, group, or packet-out keyword, correspondingly. The flow keyword may be optionally followed by one of the keywords add, modify, modify_strict, delete, or delete_strict, of which the add is assumed if a bare flow is given. Similarly, the group keyword may be optionally followed by one of the keywords add, modify, add_or_mod, delete, insert_bucket, or remove_bucket, of which the add is assumed if a bare group is given.

bundle switch file
Execute all flow and group mods in file as a single atomic transaction against switch’s tables. All bundled mods are executed in the order specified.

OpenFlow Switch Tunnel TLV Table Commands
Open vSwitch maintains a mapping table between tunnel option TLVs (defined by <class, type, length>) and NXM fields tun_metadata , where n ranges from 0 to 63, that can be operated on for the purposes of matches, actions, etc. This TLV table can be used for Geneve option TLVs or other protocols with options in same TLV format as Geneve options. This mapping must be explicitly specified by the user through the following commands.

A TLV mapping is specified with the syntax {class=class,type=type,len=length}->tun_metadata . When an option mapping exists for a given tun_metadata , matching on the defined field becomes possible, e.g.:

ovs-ofctl add-tlv-map br0 "{class=0xffff,type=0,len=4}->tun_metadata0"

ovs-ofctl add-flow br0 tun_metadata0=1234,actions=controller

A mapping should not be changed while it is in active use by a flow. The result of doing so is undefined.

These commands are Nicira extensions to OpenFlow and require Open vSwitch 2.5 or later.

add-tlv-map switch option[option]...
Add each option to switch’s tables. Duplicate fields are rejected.

del-tlv-map switch [option[option]]...
Delete each option from switch’s table, or all option TLV mapping if no option is specified. Fields that aren’t mapped are ignored.

dump-tlv-map switch
Show the currently mapped fields in the switch’s option table as well as switch capabilities.

OpenFlow Switch Monitoring Commands
snoop switch
Connects to switch and prints to the console all OpenFlow messages received. Unlike other ovs-ofctl commands, if switch is the name of a bridge, then the snoop command connects to a Unix domain socket named /usr/local/var/run/openvswitch/switch.snoop. ovs-vswitchd listens on such a socket for each bridge and sends to it all of the OpenFlow messages sent to or received from its configured OpenFlow controller. Thus, this command can be used to view OpenFlow protocol activity between a switch and its controller.

When a switch has more than one controller configured, only the traffic to and from a single controller is output. If none of the controllers is configured as a primary or a secondary (using a
Nicira extension to OpenFlow 1.0 or 1.1, or a standard request in OpenFlow 1.2 or later), then a controller is chosen arbitrarily among them. If there is a primary controller, it is chosen; otherwise, if there are any controllers that are not primaries or secondaries, one is chosen arbitrarily; otherwise, a secondary controller is chosen arbitrarily. This choice is made once at connection time and does not change as controllers reconfigure their roles.

If a switch has no controller configured, or if the configured controller is disconnected, no traffic is sent, so monitoring will not show any traffic.

```
monitor switch [miss-len] [invalid_ttl] [watch:[spec...]]
```

Connects to `switch` and prints to the console all OpenFlow messages received. Usually, `switch` should specify the name of a bridge in the `ovs-vswitchd` database. This is available only in OpenFlow 1.0 as Nicira extension.

If `miss-len` is provided, `ovs-ofctl` sends an OpenFlow “set configuration” message at connection setup time that requests `miss-len` bytes of each packet that misses the flow table. Open vSwitch does not send these and other asynchronous messages to an `ovs-ofctl monitor` client connection unless a nonzero value is specified on this argument. (Thus, if `miss-len` is not specified, very little traffic will ordinarily be printed.)

If `invalid_ttl` is passed, `ovs-ofctl` sends an OpenFlow “set configuration” message at connection setup time that requests `INVALID_TTL_TO_CONTROLLER`, so that `ovs-ofctl monitor` can receive “packet-in” messages when TTL reaches zero on `dec_ttl` action. Only OpenFlow 1.1 and 1.2 support `invalid_ttl`; Open vSwitch also implements it for OpenFlow 1.0 as an extension.

`watch:[spec...]` causes `ovs-ofctl` to send a “monitor request” Nicira extension message to the switch at connection setup time. This message causes the switch to send information about flow table changes as they occur. The following comma-separated `spec` syntax is available:

- `!initial` Do not report the switch’s initial flow table contents.
- `!add` Do not report newly added flows.
- `!delete` Do not report deleted flows.
- `!modify` Do not report modifications to existing flows.
- `!own` Abbreviate changes made to the flow table by `ovs-ofctl`’s own connection to the switch. (These could only occur using the `ofctl/send` command described below under RUN-TIME MANAGEMENT COMMANDS.)
- `!actions` Do not report actions as part of flow updates.
- `table=table` Limits the monitoring to the table with the given `table`, which may be expressed as a number between 0 and 254 or (unless `--no-names` is specified) a name. By default, all tables are monitored.
- `out_port=port` If set, only flows that output to `port` are monitored. The `port` may be an OpenFlow port number or keyword (e.g. `LOCAL`).
- `field=value` Monitors only flows that have `field` specified as the given `value`. Any syntax valid for matching on `dump-flows` may be used.

This command may be useful for debugging switch or controller implementations. With `watch:`, it is particularly useful for observing how a controller updates flow tables.

**OpenFlow Switch and Controller Commands**

The following commands, like those in the previous section, may be applied to OpenFlow switches, using any of the connection methods described in that section. Unlike those commands, these may also be
applied to OpenFlow controllers.

probe target
Sends a single OpenFlow echo-request message to target and waits for the response. With the \(-t\) or \(--timeout\) option, this command can test whether an OpenFlow switch or controller is up and running.

ping target [n]
Sends a series of 10 echo request packets to target and times each reply. The echo request packets consist of an OpenFlow header plus \(n\) bytes (default: 64) of randomly generated payload. This measures the latency of individual requests.

benchmark target n count
Sends \(n\) echo request packets that each consist of an OpenFlow header plus \(n\) bytes of payload and waits for each response. Reports the total time required. This is a measure of the maximum bandwidth to target for round-trips of \(n\)-byte messages.

Other Commands
ofp-parse file
Reads file (or stdin if file is \(\sim\)) as a series of OpenFlow messages in the binary format used on an OpenFlow connection, and prints them to the console. This can be useful for printing OpenFlow messages captured from a TCP stream.

ofp-parse-pcap file [port...]
Reads file, which must be in the PCAP format used by network capture tools such as tcpdump or wireshark, extracts all the TCP streams for OpenFlow connections, and prints the OpenFlow messages in those connections in human-readable format on stdout.

OpenFlow connections are distinguished by TCP port number. Non-OpenFlow packets are ignored. By default, data on TCP ports 6633 and 6653 are considered to be OpenFlow. Specify one or more port arguments to override the default.

This command cannot usefully print SSL encrypted traffic. It does not understand IPv6.

Flow Syntax
Some ovs-ofctl commands accept an argument that describes a flow or flows. Such flow descriptions comprise a series of field=value assignments, separated by commas or white space. (Embedding spaces into a flow description normally requires quoting to prevent the shell from breaking the description into multiple arguments.)

Flow descriptions should be in normal form. This means that a flow may only specify a value for an L3 field if it also specifies a particular L2 protocol, and that a flow may only specify an L4 field if it also specifies particular L2 and L3 protocol types. For example, if the L2 protocol type dl_type is wildcarded, then L3 fields nw_src, nw_dst, and nw_proto must also be wildcarded. Similarly, if dl_type or nw_proto (the L3 protocol type) is wildcarded, so must be the L4 fields tcp_dst and tcp_src. ovs-ofctl will warn about flows not in normal form.

ovs-fields(7) describes the supported fields and how to match them. In addition to match fields, commands that operate on flows accept a few additional key-value pairs:

\[table=table\]
For flow dump commands, limits the flows dumped to those in table, which may be expressed as a number between 0 and 255 or (unless \(--no-names\) is specified) a name. If not specified (or if 255 is specified as table), then flows in all tables are dumped.

For flow table modification commands, behavior varies based on the OpenFlow version used to connect to the switch:

OpenFlow 1.0
OpenFlow 1.0 does not support table for modifying flows. ovs-ofctl will exit with an error if table (other than table=255) is specified for a switch that only supports OpenFlow 1.0.
In OpenFlow 1.0, the switch chooses the table into which to insert a new flow. The Open vSwitch software switch always chooses table 0. Other Open vSwitch datapaths and other OpenFlow implementations may choose different tables.

The OpenFlow 1.0 behavior in Open vSwitch for modifying or removing flows depends on whether \texttt{--strict} is used. Without \texttt{--strict}, the command applies to matching flows in all tables. With \texttt{--strict}, the command will operate on any single matching flow in any table; it will do nothing if there are matches in more than one table. (The distinction between these behaviors only matters if non-OpenFlow 1.0 commands were also used, because OpenFlow 1.0 alone cannot add flows with the same matching criteria to multiple tables.)

OpenFlow 1.0 with table\_id extension

Open vSwitch implements an OpenFlow extension that allows the controller to specify the table on which to operate. \texttt{ovs-ofctl} automatically enables the extension when \texttt{table} is specified and OpenFlow 1.0 is used. \texttt{ovs-ofctl} automatically detects whether the switch supports the extension. As of this writing, this extension is only known to be implemented by Open vSwitch.

With this extension, \texttt{ovs-ofctl} operates on the requested table when \texttt{table} is specified, and acts as described for OpenFlow 1.0 above when no \texttt{table} is specified (or for \texttt{table=255}).

OpenFlow 1.1

OpenFlow 1.1 requires flow table modification commands to specify a table. When \texttt{table} is not specified (or \texttt{table=255} is specified), \texttt{ovs-ofctl} defaults to table 0.

OpenFlow 1.2 and later

OpenFlow 1.2 and later allow flow deletion commands, but not other flow table modification commands, to operate on all flow tables, with the behavior described above for OpenFlow 1.0.

\texttt{duration=...}
\texttt{n\_packet=...}
\texttt{n\_bytes=...}

\texttt{ovs-ofctl} ignores assignments to these “fields” to allow output from the \texttt{dump-flows} command to be used as input for other commands that parse flows.

The \texttt{add-flows}, \texttt{add-flows}, and \texttt{mod-flows} commands require an additional field, which must be the final field specified:

\texttt{actions=[action][action...]}

Specifies a comma-separated list of actions to take on a packet when the flow entry matches. If no \texttt{action} is specified, then packets matching the flow are dropped. See \texttt{ovs-actions(7)} for details on the syntax and semantics of actions.

An opaque identifier called a cookie can be used as a handle to identify a set of flows:

\texttt{cookie=}

A cookie can be associated with a flow using the \texttt{add-flows}, \texttt{add-flows}, and \texttt{mod-flows} commands. \texttt{value} can be any 64-bit number and need not be unique among flows. If this field is omitted, a default cookie value of 0 is used.

\texttt{cookie=}

When using NXM, the cookie can be used as a handle for querying, modifying, and deleting flows. \texttt{value} and \texttt{mask} may be supplied for the \texttt{del-flows}, \texttt{mod-flows}, \texttt{dump-flows}, and \texttt{dump-aggregate} commands to limit matching cookies. A 1-bit in \texttt{mask} indicates that the corresponding bit in \texttt{cookie} must match exactly, and a 0-bit wildcards that bit. A mask of −1 may be used to exactly match a cookie.

The \texttt{mod-flows} command can update the cookies of flows that match a cookie by specifying the \texttt{cookie} field twice (once with a mask for matching and once without to indicate the new value):
ovs−ofctl mod−flows br0 cookie=1,actions=normal
Change all flows’ cookies to 1 and change their actions to normal.

ovs−ofctl mod−flows br0 cookie=1/−1,cookie=2,actions=normal
Update cookies with a value of 1 to 2 and change their actions to normal.

The ability to match on cookies was added in Open vSwitch 1.5.0.

The following additional field sets the priority for flows added by the add−flow and add−flows commands. For mod−flows and del−flows when −−strict is specified, priority must match along with the rest of the flow specification. For mod−flows without −−strict, priority is only significant if the command creates a new flow, that is, non-strict mod−flows does not match on priority and will not change the priority of existing flows. Other commands do not allow priority to be specified.

**priority=value**

The priority at which a wildcarded entry will match in comparison to others. value is a number between 0 and 65535, inclusive. A higher value will match before a lower one. An exact-match entry will always have priority over an entry containing wildcards, so it has an implicit priority value of 65535. When adding a flow, if the field is not specified, the flow’s priority will default to 32768.

OpenFlow leaves behavior undefined when two or more flows with the same priority can match a single packet. Some users expect “sensible” behavior, such as more specific flows taking precedence over less specific flows, but OpenFlow does not specify this and Open vSwitch does not implement it. Users should therefore take care to use priorities to ensure the behavior that they expect.

The add−flow, add−flows, and mod−flows commands support the following additional options. These options affect only new flows. Thus, for add−flow and add−flows, these options are always significant, but for mod−flows they are significant only if the command creates a new flow, that is, their values do not update or affect existing flows.

**idle_timeout=seconds**
Causes the flow to expire after the given number of seconds of inactivity. A value of 0 (the default) prevents a flow from expiring due to inactivity.

**hard_timeout=seconds**
Causes the flow to expire after the given number of seconds, regardless of activity. A value of 0 (the default) gives the flow no hard expiration deadline.

**importance=value**
Sets the importance of a flow. The flow entry eviction mechanism can use importance as a factor in deciding which flow to evict. A value of 0 (the default) makes the flow non-evictable on the basis of importance. Specify a value between 0 and 65535.

Only OpenFlow 1.4 and later support importance.

**send_flow_rem**
Marks the flow with a flag that causes the switch to generate a “flow removed” message and send it to interested controllers when the flow later expires or is removed.

**check_overlap**
Forces the switch to check that the flow match does not overlap that of any different flow with the same priority in the same table. (This check is expensive so it is best to avoid it.)

**reset_counts**
When this flag is specified on a flow being added to a switch, and the switch already has a flow with an identical match, an OpenFlow 1.2 (or later) switch resets the flow’s packet and byte counters to 0. Without the flag, the packet and byte counters are preserved.

OpenFlow 1.0 and 1.1 switches always reset counters in this situation, as if reset_counts were always specified.
Open vSwitch 1.10 added support for **reset_counts**.

**no_packet_counts**

**no_byte_counts**

Adding these flags to a flow advises an OpenFlow 1.3 (or later) switch that the controller does not need packet or byte counters, respectively, for the flow. Some switch implementations might achieve higher performance or reduce resource consumption when these flags are used. These flags provide no benefit to the Open vSwitch software switch implementation.

OpenFlow 1.2 and earlier do not support these flags.

Open vSwitch 1.10 added support for **no_packet_counts** and **no_byte_counts**.

The **dump−flows**, **dump−aggregate**, **del−flow** and **del−flows** commands support these additional optional fields:

**out_port=port**

If set, a matching flow must include an output action to port, which must be an OpenFlow port number or name (e.g. local).

**out_group=group**

If set, a matching flow must include an group action naming group, which must be an OpenFlow group number. This field is supported in Open vSwitch 2.5 and later and requires OpenFlow 1.1 or later.

**Table Entry Output**

The **dump−tables** and **dump−aggregate** commands print information about the entries in a datapath’s tables. Each line of output is a flow entry as described in **Flow Syntax**, above, plus some additional fields:

**duration=secs**

The time, in seconds, that the entry has been in the table. secs includes as much precision as the switch provides, possibly to nanosecond resolution.

**n_packets**

The number of packets that have matched the entry.

**n_bytes**

The total number of bytes from packets that have matched the entry.

The following additional fields are included only if the switch is Open vSwitch 1.6 or later and the NXM flow format is used to dump the flow (see the description of the **−−flow-format** option below). The values of these additional fields are approximations only and in particular **idle_age** will sometimes become non-zero even for busy flows.

**hard_age=secs**

The integer number of seconds since the flow was added or modified. hard_age is displayed only if it differs from the integer part of duration. (This is separate from duration because mod−flows restarts the hard_timeout timer without zeroing duration.)

**idle_age=secs**

The integer number of seconds that have passed without any packets passing through the flow.

**Packet−Out Syntax**

**ovs−ofctl bundle** command accepts packet-outs to be specified in the bundle file. Each packet-out comprises of a series of **field=value** assignments, separated by commas or white space. (Embedding spaces into a packet-out description normally requires quoting to prevent the shell from breaking the description into multiple arguments.). Unless noted otherwise only the last instance of each field is honoured. This same syntax is also supported by the **ovs−ofctl packet-out** command.

**in_port=port**

The port number to be considered the in_port when processing actions. This can be any valid OpenFlow port number, or any of the LOCAL, CONTROLLER, or NONE. This field is required.
pipeline_field=value

Optionally, user can specify a list of pipeline fields for a packet-out message. The supported pipeline fields includes tunnel fields and register fields as defined in ovs–fields(7).

packet=hex-string

The actual packet to send, expressed as a string of hexadecimal bytes. This field is required.

actions= [action][, action...]

The syntax of actions are identical to the actions= field described in Flow Syntax above. Specifying actions= is optional, but omitting actions is interpreted as a drop, so the packet will not be sent anywhere from the switch. actions must be specified at the end of each line, like for flow mods.

Group Syntax

Some ovs–ofctl commands accept an argument that describes a group or groups. Such flow descriptions comprise a series field=value assignments, separated by commas or white space. (Embedding spaces into a group description normally requires quoting to prevent the shell from breaking the description into multiple arguments.). Unless noted otherwise only the last instance of each field is honoured.

group_id=id

The integer group id of group. When this field is specified in del–groups or dump–groups, the keyword "all" may be used to designate all groups. This field is required.

type=type

The type of the group. The add-group, add-groups and mod-groups commands require this field. It is prohibited for other commands. The following keywords designated the allowed types:

all Execute all buckets in the group.
select Execute one bucket in the group, balancing across the buckets according to their weights.
To select a bucket, for each live bucket, Open vSwitch hashes flow data with the bucket ID and multiplies by the bucket weight to obtain a “score,” and then selects the bucket with the highest score. Use selection_method to control the flow data used for selection.
indirect
Executes the one bucket in the group.
ff
fast_failover
Executes the first live bucket in the group which is associated with a live port or group.

command_bucket_id=id

The bucket to operate on. The insert-buckets and remove-buckets commands require this field. It is prohibited for other commands. id may be an integer or one of the following keywords:

all Operate on all buckets in the group. Only valid when used with the remove-buckets command in which case the effect is to remove all buckets from the group.
first Operate on the first bucket present in the group. In the case of the insert-buckets command the effect is to insert new buckets just before the first bucket already present in the group; or to replace the buckets of the group if there are no buckets already present in the group. In the case of the remove-buckets command the effect is to remove the first bucket of the group; or do nothing if there are no buckets present in the group.
last Operate on the last bucket present in the group. In the case of the insert-buckets command the effect is to insert new buckets just after the last bucket already present in the group; or to replace the buckets of the group if there are no buckets already present in the group. In the case of the remove-buckets command the effect is to remove the last
bucket of the group; or do nothing if there are no buckets present in the group.

If `id` is an integer then it should correspond to the `bucket_id` of a bucket present in the group. In case of the `insert-buckets` command the effect is to insert buckets just before the bucket in the group whose `bucket_id` is `id`. In case of the `remove-buckets` command the effect is to remove the in the group whose `bucket_id` is `id`. It is an error if there is no bucket present group in whose `bucket_id` is `id`.

`selection_method=method`

The selection method used to select a bucket for a select group. This is a string of 1 to 15 bytes in length known to lower layers. This field is optional for `add-group`, `add-groups` and `mod-group` commands on groups of type `select`. Prohibited otherwise. If no selection method is specified, Open vSwitch up to release 2.9 applies the `hash` method with default fields. From 2.10 onwards Open vSwitch defaults to the `dp_hash` method with symmetric L3/L4 hash algorithm, as long as the weighted group buckets can be mapped to `dp_hash` values with sufficient accuracy. In 2.10 this was restricted to a maximum of 64 buckets, and in 2.17 the limit was raised to 256 buckets. In those rare cases Open vSwitch 2.10 and later fall back to the `hash` method with the default set of hash fields.

`dp_hash`

Use a datapath computed hash value. The hash algorithm varies across different datapath implementations. `dp_hash` uses the upper 32 bits of the `selection_method_param` as the datapath hash algorithm selector. The supported values are 0 (corresponding to hash computation over the IP 5-tuple) and 1 (corresponding to a symmetric hash computation over the IP 5-tuple). Selecting specific fields with the `fields` option is not supported with `dp_hash`). The lower 32 bits are used as the hash basis.

Using `dp_hash` has the advantage that it does not require the generated datapath flows to exact match any additional packet header fields. For example, even if multiple TCP connections thus hashed to different select group buckets have different source port numbers, generally all of them would be handled with a small set of already established datapath flows, resulting in less latency for TCP SYN packets. The downside is that the shared datapath flows must match each packet twice, as the datapath hash value calculation happens only when needed, and a second match is required to match some bits of its value. This double-matching incurs a small additional latency cost for each packet, but this latency is orders of magnitude less than the latency of creating new datapath flows for new TCP connections.

`hash`

Use a hash computed over the fields specified with the `fields` option, see below. If no hash fields are specified, `hash` defaults to a symmetric hash over the combination of MAC addresses, VLAN tags, Ether type, IP addresses and L4 port numbers. `hash` uses the `selection_method_param` as the hash basis.

Note that the hashed fields become exact matched by the datapath flows. For example, if the TCP source port is hashed, the created datapath flows will match the specific TCP source port value present in the packet received. Since each TCP connection generally has a different source port value, a separate datapath flow will be need to be inserted for each TCP connection thus hashed to a select group bucket.

This option uses a Netronome OpenFlow extension which is only supported when using Open vSwitch 2.4 and later with OpenFlow 1.5 and later.

`selection_method_param=param`

64-bit integer parameter to the selection method selected by the `selection_method` field. The parameter’s use is defined by the lower-layer that implements the `selection_method`. It is optional if the `selection_method` field is specified as a non-empty string. Prohibited otherwise. The default value is zero.
This option uses a Netronome OpenFlow extension which is only supported when using Open vSwitch 2.4 and later with OpenFlow 1.5 and later.

```
fields=field
fields(field[=mask[...])
```

The field parameters to selection method selected by the `selection_method` field. The syntax is described in Flow Syntax with the additional restrictions that if a value is provided it is treated as a wildcard mask and wildcard masks following a slash are prohibited. The pre-requisites of fields must be provided by any flows that output to the group. The use of the fields is defined by the lower-layer that implements the `selection_method`. They are optional if the `selection_method` field is specified as ‘hash’, prohibited otherwise. The default is no fields.

This option will use a Netronome OpenFlow extension which is only supported when using Open vSwitch 2.4 and later with OpenFlow 1.5 and later.

```
bucket=bucket_parameters
```

The `add-group`, `add-groups` and `mod-group` commands require at least one bucket field. Bucket fields must appear after all other fields. Multiple bucket fields to specify multiple buckets. The order in which buckets are specified corresponds to their order in the group. If the type of the group is "indirect" then only one group may be specified. `bucket_parameters` consists of a list of `field=value` assignments, separated by commas or white space followed by a comma-separated list of actions. The fields for `bucket_parameters` are:

```
bucket_id=id
```

The 32-bit integer group id of the bucket. Values greater than 0xfffff00 are reserved. This field was added in Open vSwitch 2.4 to conform with the OpenFlow 1.5 specification. It is not supported when earlier versions of OpenFlow are used. Open vSwitch will automatically allocate bucket ids when they are not specified.

```
actions=[action[,"action...
```

The syntax of actions are identical to the `actions=` field described in Flow Syntax above. Specifying `actions=` is optional, any unknown bucket parameter will be interpreted as an action.

```
weight=value
```

The relative weight of the bucket as an integer. This may be used by the switch during bucket select for groups whose `type` is `select`.

```
watch_port=port
```

Port used to determine liveness of group. This or the `watch_group` field is required for groups whose `type` is `ff` or `fast_failover`. This or the `watch_group` field can also be used for groups whose `type` is `select`.

```
watch_group=group_id
```

Group identifier of group used to determine liveness of group. This or the `watch_port` field is required for groups whose `type` is `ff` or `fast_failover`. This or the `watch_port` field can also be used for groups whose `type` is `select`.

**Meter Syntax**

The meter table commands accept an argument that describes a meter. Such meter descriptions comprise a series `field=value` assignments, separated by commas or white space. (Embedding spaces into a group description normally requires quoting to prevent the shell from breaking the description into multiple arguments.). Unless noted otherwise only the last instance of each field is honoured.

```
meter=id
```

The identifier for the meter. An integer is used to specify a user-defined meter. In addition, the keywords "all", "controller", and "slowpath", are also supported as virtual meters. The "controller" and "slowpath" virtual meters apply to packets sent to the controller and to the OVS
userspace, respectively.

When this field is specified in `del-meter`, `dump-meter`, or `meter-stats`, the keyword "all" may be used to designate all meters. This field is required, except for `meter-stats`, which dumps all stats when this field is not specified.

- **kbps**: The unit for the `rate` and `burst_size` band parameters. `kbps` specifies kilobits per second, and `pktps` specifies packets per second. A unit is required for the `add-meter` and `mod-meter` commands.
- **pktps**: The unit for the `rate` and `burst_size` band parameters. `kbps` specifies kilobits per second, and `pktps` specifies packets per second. A unit is required for the `add-meter` and `mod-meter` commands.

- **burst**: If set, enables burst support for meter bands through the `burst_size` parameter.

- **stats**: If set, enables the collection of meter and band statistics.

- **bands=band_parameters**: The `add-meter` and `mod-meter` commands require at least one band specification. Bands must appear after all other fields.

  - **type=type**: The type of the meter band. This keyword starts a new band specification. Each band specifies a rate above which the band is to take some action. The action depends on the band type. If multiple bands’ rate is exceeded, then the band with the highest rate among the exceeded bands is selected. The following keywords designate the allowed meter band types:
    - **drop**: Drop packets exceeding the band’s rate limit.

The other `band_parameters` are:

- **rate=value**: The relative rate limit for this band, in kilobits per second or packets per second, depending on whether `kbps` or `pktps` was specified.

- **burst_size=size**: If `burst` is specified for the meter entry, configures the maximum burst allowed for the band in kilobits or packets, depending on whether `kbps` or `pktps` was specified. If unspecified, the switch is free to select some reasonable value depending on its configuration.

**OPTIONS**

- **--strict**: Uses strict matching when running flow modification commands.

- **--names**
  - **--no-names**: Every OpenFlow port has a name and a number, and every OpenFlow flow table has a number and sometimes a name. By default, `ovs-ofctl` commands accept both port and table names and numbers, and they display port and table names if `ovs-ofctl` is running on an interactive console, numbers otherwise. With **--names**, `ovs-ofctl` commands both accept and display port and table names; with **--no-names**, commands neither accept nor display port and table names.

If a port or table name contains special characters or might be confused with a keyword within a flow, it may be enclosed in double quotes (escaped from the shell). If necessary, JSON-style escape sequences may be used inside quotes, as specified in RFC 7159. When it displays port and table names, `ovs-ofctl` quotes any name that does not start with a letter followed by letters or digits.

Open vSwitch added support for port names and these options. Open vSwitch 2.10 added support for table names. Earlier versions always behaved as if **--no-names** were specified.
Open vSwitch does not place its own limit on the length of port names, but OpenFlow limits port names to 15 bytes. Because ovs-ofctl uses OpenFlow to retrieve the mapping between port names and numbers, names longer than this limit will be truncated for both display and acceptance. Truncation can also cause long names that are different to appear to be the same; when a switch has two ports with the same (truncated) name, ovs-ofctl refuses to display or accept the name, using the number instead.

OpenFlow and Open vSwitch limit table names to 32 bytes.

--stats
--no-stats

The dump-flows command by default, or with --stats, includes flow duration, packet and byte counts, and idle and hard age in its output. With --no-stats, it omits all of these, as well as cookie values and table IDs if they are zero.

--read-only

Do not execute read/write commands.

--bundle

Execute flow mods as an OpenFlow 1.4 atomic bundle transaction.

• Within a bundle, all flow mods are processed in the order they appear and as a single atomic transaction, meaning that if one of them fails, the whole transaction fails and none of the changes are made to the switch’s flow table, and that each given datapath packet traversing the OpenFlow tables sees the flow tables either as before the transaction, or after all the flow mods in the bundle have been successfully applied.

• The beginning and the end of the flow table modification commands in a bundle are delimited with OpenFlow 1.4 bundle control messages, which makes it possible to stream the included commands without explicit OpenFlow barriers, which are otherwise used after each flow table modification command. This may make large modifications execute faster as a bundle.

• Bundles require OpenFlow 1.4 or higher. An explicit -O OpenFlow14 option is not needed, but you may need to enable OpenFlow 1.4 support for OVS by setting the OVSDB protocols column in the bridge table.

-O [version,version,...]
--protocols=[version,version,...]

Sets the OpenFlow protocol versions that are allowed when establishing an OpenFlow session. These protocol versions are enabled by default:

• OpenFlow10, for OpenFlow 1.0.

The following protocol versions are generally supported, but for compatibility with older versions of Open vSwitch they are not enabled by default:

• OpenFlow11, for OpenFlow 1.1.
• OpenFlow12, for OpenFlow 1.2.
• OpenFlow13, for OpenFlow 1.3.
• OpenFlow14, for OpenFlow 1.4.
• OpenFlow15, for OpenFlow 1.5.

-F format,format...
--flow-format=format,format...

ovs-ofctl supports the following individual flow formats, any number of which may be listed as format:

OpenFlow10--table_id

This is the standard OpenFlow 1.0 flow format. All OpenFlow switches and all versions of Open vSwitch support this flow format.
OpenFlow10+table_id
This is the standard OpenFlow 1.0 flow format plus a Nicira extension that allows
ovs-ofctl to specify the flow table in which a particular flow should be placed. Open
vSwitch 1.2 and later supports this flow format.

NXM–table_id (Nicira Extended Match)
This Nicira extension to OpenFlow is flexible and extensible. It supports all of the Nicira
flow extensions, such as tun_id and registers. Open vSwitch 1.1 and later supports this
flow format.

NXM+table_id (Nicira Extended Match)
This combines Nicira Extended match with the ability to place a flow in a specific table.
Open vSwitch 1.2 and later supports this flow format.

OXM-OpenFlow12
OXM-OpenFlow13
OXM-OpenFlow14
OXM-OpenFlow15
These are the standard OXM (OpenFlow Extensible Match) flow format in OpenFlow 1.2
and later.

ovs-ofctl also supports the following abbreviations for collections of flow formats:
any Any supported flow format.

OpenFlow10
OpenFlow10–table_id or OpenFlow10+table_id.

NXM NXM–table_id or NXM+table_id.

OXM OXM-OpenFlow12, OXM-OpenFlow13, or OXM-OpenFlow14.

For commands that modify the flow table, ovs-ofctl by default negotiates the most widely sup-
ported flow format that supports the flows being added. For commands that query the flow table,
ovs-ofctl by default uses the most advanced format supported by the switch.

This option, where format is a comma-separated list of one or more of the formats listed above,
limits ovs-ofctl’s choice of flow format. If a command cannot work as requested using one of the
specified flow formats, ovs-ofctl will report a fatal error.

-P format
---packet–in–format=format

ovs-ofctl supports the following “packet-in” formats, in order of increasing capability:

standard
This uses the OFPT_PACKET_IN message, the standard “packet-in” message for any
given OpenFlow version. Every OpenFlow switch that supports a given OpenFlow ver-
sion supports this format.

nxt_packet_in
This uses the NXT_PACKET_IN message, which adds many of the capabilities of the
OpenFlow 1.1 and later “packet-in” messages before those OpenFlow versions were
available in Open vSwitch. Open vSwitch 1.1 and later support this format. Only Open
vSwitch 2.6 and later, however, support it for OpenFlow 1.1 and later (but there is little
reason to use it with those versions of OpenFlow).

nxt_packet_in2
This uses the NXT_PACKET_IN2 message, which is extensible and should avoid the
need to define new formats later. In particular, this format supports passing arbitrary
user-provided data to a controller using the userdata option on the controller action.
Open vSwitch 2.6 and later support this format.
Without this option, `ovs-ofctl` prefers `nxt_packet_in2` if the switch supports it. Otherwise, if OpenFlow 1.0 is in use, `ovs-ofctl` prefers `nxt_packet_in` if the switch supports it. Otherwise, `ovs-ofctl` falls back to the standard packet-in format. When this option is specified, `ovs-ofctl` insists on the selected format. If the switch does not support the requested format, `ovs-ofctl` will report a fatal error.

Before version 2.6, Open vSwitch called standard format `openflow10` and `nxt_packet_in` format `nxm`, and `ovs-ofctl` still accepts these names as synonyms. (The name `openflow10` was a misnomer because this format actually varies from one OpenFlow version to another; it is not consistently OpenFlow 1.0 format. Similarly, when `nxt_packet_in2` was introduced, the name `nxm` became confusing because it also uses OXM/NXM.)

This option affects only the `monitor` command.

`--timestamp`

Print a timestamp before each received packet. This option only affects the `monitor`, `snoop`, and `ofp--parse--pcap` commands.

`--more`

Increases the verbosity of OpenFlow messages printed and logged by `ovs-ofctl` commands. Specify this option more than once to increase verbosity further.

`--sort[=field]`

`--rsort[=field]`

Display output sorted by flow `field` in ascending (`--sort`) or descending (`--rsort`) order, where `field` is any of the fields that are allowed for matching or `priority` to sort by priority. When `field` is omitted, the output is sorted by priority. Specify these options multiple times to sort by multiple fields.

Any given flow will not necessarily specify a value for a given field. This requires special treatment:

- A flow that does not specify any part of a field that is used for sorting is sorted after all the flows that do specify the field. For example, `--sort(tcp_src)` will sort all the flows that specify a TCP source port in ascending order, followed by the flows that do not specify a TCP source port at all.
- A flow that only specifies some bits in a field is sorted as if the wildcarded bits were zero. For example, `--sort=nw_src` would sort a flow that specifies `nw_src=192.168.0.0/24` the same as `nw_src=192.168.0.0`.

These options currently affect only `dump--flows` output.

**Daemon Options**

The following options are valid on POSIX based platforms.

`--pidfile[=pidfile]`

Causes a file (by default, `ovs-ofctl.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, `ovs-ofctl` 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 ovs-ofctl 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. ovs-ofctl detaches only when executing the monitor or snoop commands.

--monitor
Creates an additional process to monitor the ovs-ofctl 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, ovs-ofctl changes its current working directory to the root directory after it detaches. Otherwise, invoking ovs-ofctl 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 ovs-ofctl 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 ovs-ofctl 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.

--unixctl=socket
Sets the name of the control socket on which ovs-ofctl listens for runtime management commands (see RUNTIME MANAGEMENT COMMANDS, below). If socket does not begin with /, it is interpreted as relative to /usr/local/var/run/openvswitch. If --unixctl is not used at all, the default socket is /usr/local/var/run/openvswitch/ovs-ofctl.pid.ctl, where pid is ovs-ofctl’s process ID.

On Windows a local named pipe is used to listen for runtime management commands. A file is created in the absolute path as pointed by socket or if --unixctl is not used at all, a file is created as ovs-ofctl.ctl in the configured OVS_RUNDIR directory. The file exists just to mimic the behavior of a Unix domain socket.

Specifying none for socket disables the control socket feature.
Public Key Infrastructure Options

```
-p privkey.pem
--private-key=privkey.pem
```
Specifies a PEM file containing the private key used as ovs-ofctl’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 ovs-ofctl 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.

```
-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-ofctl closes its standard file descriptors, so logging to the console will have no effect.)
- off, emerg, 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.

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).

- 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-ofctl.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.

--color[=when]
Colorize the output (for some commands); when can be never, always, or auto (the default).

Only some commands support output coloring. Color names and default colors may change in future releases.

The environment variable OVS_COLORS can be used to specify user-defined colors and other attributes used to highlight various parts of the output. If set, its value is a colon-separated list of capabilities that defaults to ac:01;31:dr=34:le=31:pm=36:pr=35:sp=33:vl=32. Supported capabilities were initially designed for coloring flows from ovs-ofctl dump-flows switch command, and they are as follows.

ac=01;31
SGR substring for actions= keyword in a flow. The default is a bold red text foreground.

dr=34 SGR substring for drop keyword. The default is a dark blue text foreground.

le=31 SGR substring for learn= keyword in a flow. The default is a red text foreground.
pm=36 SGR substring for flow match attribute names. The default is a cyan text foreground.

pr=35 SGR substring for keywords in a flow that are followed by arguments inside parenthesis. The default is a magenta text foreground.

sp=33 SGR substring for some special keywords in a flow, notably: table=, priority=, load=, output=, move=, group=, CONTROLLER=, set_field=, resubmit=, exit. The default is a yellow text foreground.

vl=32 SGR substring for a lone flow match attribute with no field name. The default is a green text foreground.

See the Select Graphic Rendition (SGR) section in the documentation of the text terminal that is used for permitted values and their meaning as character attributes.

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

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

**RUNTIME MANAGEMENT COMMANDS**

**ovs-appctl(8)** can send commands to a running **ovs-ofctl** process. The supported commands are listed below.

**exit** Causes **ovs-ofctl** to gracefully terminate. This command applies only when executing the **monitor** or **snoop** commands.

ofctl/set-output-file file
Causes all subsequent output to go to file instead of stderr. This command applies only when executing the **monitor** or **snoop** commands.

ofctl/send ofmsg...
Sends each ofmsg, specified as a sequence of hex digits that express an OpenFlow message, on the OpenFlow connection. This command is useful only when executing the **monitor** command.

ofctl/packet-out packet-out
Sends an OpenFlow PACKET_OUT message specified in **Packet-Out Syntax**, on the OpenFlow connection. See **Packet-Out Syntax** section for more information. This command is useful only when executing the **monitor** command.

**EXAMPLES**

The following examples assume that **ovs-vswitchd** has a bridge named **br0** configured.

**ovs-ofctl dump-tables br0**
Prints out the switch’s table stats. (This is more interesting after some traffic has passed through.)

**ovs-ofctl dump-flows br0**
Prints the flow entries in the switch.

**ovs-ofctl add-flow table=0 actions=learn(table=1,hard_timeout=10, NXM_OF_VLAN_TCI[0..11],output:NNM_OF_IN_PORT[]), resubmit(,1)**

**ovs-ofctl add-flow table=1 priority=0 actions=flood** Implements a level 2 MAC learning switch using the learn.

**ovs-ofctl add-flow br0 'table=0,priority=0 actions=load:3->NRM_NX_REG0[0..15],learn(table=0,priority=1, idle.timeout=10,NXM_OF_ETH_SRC[],NRM_OF_VLAN_TCI[0..11],output:NRM_NX_REG0[0..15],output:2**

In this use of a learn action, the first packet from each source MAC will be sent to port 2.
Subsequent packets will be output to port 3, with an idle timeout of 10 seconds. NXM field names and match field names are both accepted, e.g. `NXM_NX_REG0` or `reg0` for the first register, and empty brackets may be omitted.

Additional examples may be found documented as part of related sections.

**SEE ALSO**

`ovs-fields(7)`, `ovs-actions(7)`, `ovs-appctl(8)`, `ovs-vswitchd(8)`, `ovs-vswitchd.conf.db(8)`