NAME
  ovs−dpctl − administer Open vSwitch datapaths

SYNOPSIS
  ovs−dpctl [options] command [switch] [args...]

DESCRIPTION
  The ovs−dpctl program can create, modify, and delete Open vSwitch datapaths. A single machine may host any number of datapaths.

  This program works only with datapaths that are implemented outside of ovs−vswitchd itself, such as the Linux and Windows kernel-based datapaths. To manage datapaths that are integrated into ovs−vswitchd, such as the userspace (netdev) datapath, use ovs−appctl(8) to invoke the dpctl/* commands, which are documented in ovs−vswitchd(8).

  A newly created datapath is associated with only one network device, a virtual network device sometimes called the datapath’s “local port”. A newly created datapath is not, however, associated with any of the host’s other network devices. To intercept and process traffic on a given network device, use the add−if command to explicitly add that network device to the datapath.

  If ovs−vswitchd(8) is in use, use ovs−vsctl(8) instead of ovs−dpctl.

  Most ovs−dpctl commands that work with datapaths take an argument that specifies the name of the datapath. Datapath names take the form [type@]name, where name is the network device associated with the datapath’s local port. If type is given, it specifies the datapath provider of name, otherwise the default provider system is assumed.

  The following commands manage datapaths. Do not use commands to add or remove or modify datapaths if ovs−vswitchd is running because this interferes with ovs−vswitchd’s own datapath management.

  add−dp dp [netdev[,option]...]
    Creates datapath dp, with a local port also named dp. This will fail if a network device dp already exists.

    If netdevs are specified, ovs−dpctl adds them to the new datapath, just as if add−if was specified.

  del−dp dp
    Deletes datapath dp. If dp is associated with any network devices, they are automatically removed.

  add−if dp netdev[,option]...
    Adds each netdev to the set of network devices datapath dp monitors, where dp is the name of an existing datapath, and netdev is the name of one of the host’s network devices, e.g. eth0. Once a network device has been added to a datapath, the datapath has complete ownership of the network device’s traffic and the network device appears silent to the rest of the system.

    A netdev may be followed by a comma-separated list of options. The following options are currently supported:
      type=type
        Specifies the type of port to add. The default type is system.
      port_no=port
        Requests a specific port number within the datapath. If this option is not specified then one will be automatically assigned.
      key=value
        Adds an arbitrary key-value option to the port’s configuration.

      ovs−vswitchd.conf.db(5) documents the available port types and options.

  set−if dp port[,option]...
    Reconfigures each port in dp as specified. An option of the form key=value adds the specified key-value option to the port or overrides an existing key’s value. An option of the form key=, that is, without a value, deletes the key-value named key. The type and port number of a port cannot be
changed, so `type` and `port_no` are only allowed if they match the existing configuration.

`del-if dp netdev...`
Removes each `netdev` from the list of network devices datapath `dp` monitors.

`dump-dps`
Prints the name of each configured datapath on a separate line.

`[-s | --statistics] show [dp...]`
Prints a summary of configured datapaths, including their datapath numbers and a list of ports connected to each datapath. (The local port is identified as port 0.) If `−s` or `--statistics` is specified, then packet and byte counters are also printed for each port.

The datapath numbers consists of flow stats and mega flow mask stats.

The "lookups" row displays three stats related to flow lookup triggered by processing incoming packets in the datapath. "hit" displays number of packets matches existing flows. "missed" displays the number of packets not matching any existing flow and require user space processing. "lost" displays number of packets destined for user space process but subsequently dropped before reaching userspace. The sum of "hit" and "miss" equals to the total number of packets datapath processed.

The "flows" row displays the number of flows in datapath.

The "masks" row displays the mega flow mask stats. This row is omitted for datapath not implementing mega flow. "hit" displays the total number of masks visited for matching incoming packets. "total" displays number of masks in the datapath. "hit/pkt" displays the average number of masks visited per packet; the ratio between "hit" and total number of packets processed by the datapath.

If one or more datapaths are specified, information on only those datapaths are displayed. Otherwise, `ovs-dpctl` displays information about all configured datapaths.

**DATAPATH FLOW TABLE DEBUGGING COMMANDS**

The following commands are primarily useful for debugging Open vSwitch. The flow table entries (both matches and actions) that they work with are not OpenFlow flow entries. Instead, they are different and considerably simpler flows maintained by the Open vSwitch kernel module. Do not use commands to add or remove or modify datapath flows if `ovs-vswitchd` is running because it interferes with `ovs-vswitchd`'s own datapath flow management. Use `ovs-ofctl(8)`, instead, to work with OpenFlow flow entries.

The `dp` argument to each of these commands is optional when exactly one datapath exists, in which case that datapath is the default. When multiple datapaths exist, then a datapath name is required.

`[-m | --more] [-n | --no-names] dump-flows [dp] [filter=filter] [type=type] [pmd=pmd]`
Prints to the console all flow entries in datapath `dp`'s flow table. Without `−m` or `--more`, output omits match fields that a flow wildcards entirely; with `−m` or `--more`, output includes all wildcarded fields.

If `filter=filter` is specified, only displays the flows that match the `filter`. `filter` is a flow in the form similar to that accepted by `ovs-ofctl(8)`'s `add-flow` command. (This is not an OpenFlow flow: besides other differences, it never contains wildcards.) The `filter` is also useful to match wildcarded fields in the datapath flow. As an example, `filter=’tcp,tp_src=100’` will match the datapath flow containing 'tcp(src=80/0xff00,dst=8080/0xff)'.

If `pmd=pmd` is specified, only displays flows of the specified pmd. Using `pmd=-1` will restrict the dump to flows from the main thread. This option is only supported by the `userspace datapath`.

If `type=type` is specified, only displays flows of the specified types. This option supported only for `ovs-appctl dpctl/dump-flows`. `type` is a comma separated list, which can contain any of the following:

- `ovs` - displays flows handled in the ovs dp
- `tc` - displays flows handled in the tc dp
- `dpdk` - displays flows fully offloaded by dpdk
offloaded - displays flows offloaded to the HW
non-offloaded - displays flows not offloaded to the HW
partially-offloaded - displays flows where only part of their processing is done in HW
all - displays all the types of flows

By default all the types of flows are displayed. ovs-dpctl always acts as if the type was ovs.

**add-flow [dp] flow actions**


Adds or modifies a flow in dp's flow table that, when a packet matching flow arrives, causes actions to be executed.

The *add-flow* command succeeds only if *flow* does not already exist in *dp*. Contrariwise, *mod-flow* without *--may-create* only modifies the actions for an existing flow. With *--may-create*, *mod-flow* will add a new flow or modify an existing one.

If *--s* or *--statistics* is specified, then *mod-flow* prints the modified flow’s statistics. A flow’s statistics are the number of packets and bytes that have passed through the flow, the elapsed time since the flow last processed a packet (if ever), and (for TCP flows) the union of the TCP flags processed through the flow.

With *--clear*, *mod-flow* zeros out the flow’s statistics. The statistics printed if *--s* or *--statistics* is also specified are those from just before clearing the statistics.

NOTE: *flow and actions* do not match the syntax used with ovs-ofctl(8)’s *add-flow* command.

**Usage Examples**

Forward ARP between ports 1 and 2 on datapath myDP:

```bash
ovs-dpctl add-flow myDP \
"in_port(1),eth(),eth_type(0x0806),arp()" 2
ovs-dpctl add-flow myDP \
"in_port(2),eth(),eth_type(0x0806),arp()" 1
```

Forward all IPv4 traffic between two addresses on ports 1 and 2:

```bash
ovs-dpctl add-flow myDP \
"in_port(1),eth(),eth_type(0x800),\nipv4(src=172.31.110.4,dst=172.31.110.5)" 2
ovs-dpctl add-flow myDP \
"in_port(2),eth(),eth_type(0x800),\nipv4(src=172.31.110.5,dst=172.31.110.4)" 1
```

**add-flows [dp] file**

**mod-flows [dp] file**

**del-flows [dp] file**

Reads flow entries from *file* (or *stdin* if *file* is *) and adds, modifies, or deletes each entry to the datapath. Each flow specification (e.g., each line in *file*) may start with add, modify, or delete keyword to specify whether a flow is to be added, modified, or deleted. A flow specification without one of these keywords is treated based on the used command. All flow modifications are executed as individual transactions in the order specified.


Deletes the flow from dp’s flow table that matches flow. If *--s* or *--statistics* is specified, then *del-flow* prints the deleted flow’s statistics.


Fetches the flow from dp’s flow table with unique identifier *ufid*. *ufid* must be specified as a string of 32 hexadecimal characters.
del−flows [dp]
Deletes all flow entries from datapath dp’s flow table.

DATAPATH FLOW CACHE COMMANDS

The following commands are useful for debugging and configuring the datapath flow cache settings.

cache−get−size [dp]
Prints the current cache sizes to the console.

cache−set−size [dp] cache size
Set the dp’s specific cache to the given size. The cache name can be found by using the cache−get−size command.

CONNECTION TRACKING TABLE COMMANDS

The following commands are useful for debugging and configuring the connection tracking table in the datapath.

The dp argument to each of these commands is optional when exactly one datapath exists, in which case that datapath is the default. When multiple datapaths exist, then a datapath name is required.

N.B. (Linux specific): the system datapaths (i.e. the Linux kernel module Open vSwitch datapaths) share a single connection tracking table (which is also used by other kernel subsystems, such as iptables, nftables and the regular host stack). Therefore, the following commands do not apply specifically to one datapath.

ipf−set−enabled [dp] v4|v6
ipf−set−disabled [dp] v4|v6
Enables or disables IP fragmentation handling for the userspace connection tracker. Either v4 or v6 must be specified. Both IPv4 and IPv6 fragment reassembly are enabled by default. Only supported for the userspace datapath.

ipf−set−min−frag [dp] v4|v6 minfrag
Sets the minimum fragment size (L3 header and data) for non-final fragments to minfrag. Either v4 or v6 must be specified. For enhanced DOS security, higher minimum fragment sizes can usually be used. The default IPv4 value is 1200 and the clamped minimum is 400. The default IPv6 value is 1280, with a clamped minimum of 400, for testing flexibility. The maximum fragment size is not clamped, however, setting this value too high might result in valid fragments being dropped. Only supported for userspace datapath.

ipf−set−max−nfrags [dp] maxfrags
Sets the maximum number of fragments tracked by the userspace datapath connection tracker to maxfrags. The default value is 1000 and the clamped maximum is 5000. Note that packet buffers can be held by the fragmentation module while fragments are incomplete, but will timeout after 15 seconds. Memory pool sizing should be set accordingly when fragmentation is enabled. Only supported for userspace datapath.

[-m | --more] ipf−get−status [dp]
Gets the configuration settings and fragment counters associated with the fragmentation handling of the userspace datapath connection tracker. With -m or --more, also dumps the IP fragment lists. Only supported for userspace datapath.

[-m | --more] [-s | --statistics] dump−conntrack [dp] [zone=zone]
Prints to the console all the connection entries in the tracker used by dp. If zone=zone is specified, only shows the connections in zone. With --more, some implementation specific details are included. With --statistics timeouts and timestamps are added to the output.

dump−conntrack−exp [dp] [zone=zone]
Prints to the console all the expectation entries in the tracker used by dp. If zone=zone is specified, only shows the expectations in zone. Only supported for userspace datapath.

flush−conntrack [dp] [zone=zone] [ct-origin-tuple [ct-reply-tuple]]
Flushes the connection entries in the tracker used by dp based on zone and connection tracking tuple ct-origin-tuple. If ct-tuple is not provided, flushes all the connection entries. If zone=zone is
specified, only flushes the connections in zone.

If ct-[orig|reply]-tuple is provided, flushes the connection entry specified by ct-[orig|reply]-tuple in zone. The zone defaults to 0 if it is not provided. The userspace connection tracker requires flushing with the original pre-NATed tuple and a warning log will be otherwise generated. The tuple can be partial and will remove all connections that are matching on the specified fields. In order to specify only ct-reply-tuple, provide empty string as ct-origin-tuple.

Note: Currently there is a limitation for matching on ICMP, in order to partially match on ICMP parameters the ct-[orig|reply]-tuple has to include either source or destination IP.

An example of an IPv4 ICMP ct-[orig|reply]-tuple:
"ct_nw_src=10.1.1.1,ct_nw_dst=10.1.1.2,ct_nw_proto=1,icmp_type=8,icmp_code=0,icmp_id=10"

An example of an IPv6 TCP ct-[orig|reply]-tuple:
"ct_ipv6_src=fc00::1,ct_ipv6_dst=fc00::2,ct_nw_proto=6,ct_tp_src=1,ct_tp_dst=2"

[−m | --more] ct−stats−show [dp] [zone=zone]
Displays the number of connections grouped by protocol used by dp. If zone=zone is specified, numbers refer to the connections in zone. With --more, groups by connection state for each protocol.

c−bkts [dp] gt=threshold
For each conntrack bucket, displays the number of connections used by dp. If gt=threshold is specified, bucket numbers are displayed when the number of connections in a bucket is greater than threshold.

c−set−maxconns [dp] maxconns
Sets the maximum limit of connection tracker entries to maxconns on dp. This can be used to reduce the processing load on the system due to connection tracking or simply limiting connection tracking. If the number of connections is already over the new maximum limit request then the new maximum limit will be enforced when the number of connections decreases to that limit, which normally happens due to connection expiry. Only supported for userspace datapath.

c−get−maxconns [dp]
Prints the maximum limit of connection tracker entries on dp. Only supported for userspace datapath.

c−get−nconns [dp]
Prints the current number of connection tracker entries on dp. Only supported for userspace datapath.

c−enable−tcp−seq−chk [dp]
c−disable−tcp−seq−chk [dp]
Enables or disables TCP sequence checking. When set to disabled, all sequence number verification is disabled, including for TCP resets. This is similar, but not the same as 'be_liberal' mode, as in Netfilter. Disabling sequence number verification is not an optimization in itself, but is needed for some hardware offload support which might offer some performance advantage. Sequence number checking is enabled by default to enforce better security and should only be disabled if required for hardware offload support. This command is only supported for the userspace datapath.

c−get−tcp−seq−chk [dp]
Prints whether TCP sequence checking is enabled or disabled on dp. Only supported for the userspace datapath.

c−set−sweep−interval [dp] ms
Sets the sweep interval. Only supported for the userspace datapath.
ct-get-sweep-interval [dp]
Prints the current sweep interval in ms. Only supported for the userspace datapath.

cr-set-limits [dp] [default=default_limit] [zone=zone_limit=limit]...
Sets the maximum allowed number of connections in a connection tracking zone. A specific zone
may be set to limit, and multiple zones may be specified with a comma-separated list. If a per-
zone limit for a particular zone is not specified in the datapath, it defaults to the default per-zone
limit. A default zone may be specified with the default=default_limit argument. Initially, the de-
fault per-zone limit is unlimited. An unlimited number of entries may be set with 0 limit.

cr-del-limits [dp] zone=zone[,zone]...
Deletes the connection tracking limit for zone. Multiple zones may be specified with a comma-
separated list.

cr-get-limits [dp] [zone=zone[,zone]...]
Retrieves the maximum allowed number of connections and current counts per-zone. If zone is
given, only the specified zone(s) are printed. If no zones are specified, all the zone limits and
counts are provided. The command always displays the default zone limit.

OPTIONS
  --timeout=secs
Limits ovs-dpctl runtime to approximately secs seconds. If the timeout expires, ovs-dpctl will
exit with a SIGALRM signal.

  --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 con-
     sole, or to a file, respectively. (If --detach is specified, ovs-dpctl 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 sever-
     ity 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.

  --verbose
Sets the maximum logging verbosity level, equivalent to --verbose=dbg.

  --verbose=PATTERN:destination:pattern
  --verbose=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-dpctl.log.

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

--syslog-method=method
Specify method how syslog messages should be sent to syslog daemon. Following forms are supported:

- **libc**, use libc syslog() function. Downside of using this options is that libc adds fixed prefix to every message before it is actually sent to the syslog daemon over /dev/log UNIX domain socket.

- **unix**:file, use UNIX domain socket directly. It is possible to specify arbitrary message format with this option. However, rsyslogd 8.9 and older versions use hard coded parser function anyway that limits UNIX domain socket use. If you want to use arbitrary message format with older rsyslogd versions, then use UDP socket to localhost IP address instead.

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

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

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

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

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

SEE ALSO
ovs-appctl(8), ovs-vswitchd(8)