NAME

ovs−vswitchd − Open vSwitch daemon

SYNOPSIS

ovs−vswitchd [*database*]

DESCRIPTION

A daemon that manages and controls any number of Open vSwitch switches on the local machine.

The *database* argument specifies how **ovs−vswitchd** connects to **ovsdb−server**. The default is **unix:/usr/local/var/run/db.sock**. The following forms are accepted:

ssl:*ip***:***port*
The specified SSL *port* on the host at the given *ip*, which must be expressed as an IP address (not a DNS name). The **−−private−key**, **−−certificate**, and **−−ca−cert** options are mandatory when this form is used.

tcp:*ip***:***port*
Connect to the given TCP *port* on *ip*.

unix:*file*
Connect to the Unix domain server socket named *file*.

pssl:*port*[**:***ip*]
Listen on the given SSL *port* for a connection. By default, connections are not bound to a particular local IP address, but specifying *ip* limits connections to those from the given *ip*. The **−−private−key**, **−−certificate**, and **−−ca−cert** options are mandatory when this form is used.

ptcp:*port*[**:***ip*]
Listen on the given TCP *port* for a connection. By default, connections are not bound to a particular local IP address, but *ip* may be specified to listen only for connections to the given *ip*.

punix:*file*
Listen on the Unix domain server socket named *file* for a connection.

ovs−vswitchd retrieves its configuration from *database* at startup. It sets up Open vSwitch datapaths and then operates switching across each bridge described in its configuration files. As the database changes, **ovs−vswitchd** automatically updates its configuration to match.

Upon receipt of a SIGHUP signal, **ovs−vswitchd** reopens its log file, if one was specified on the command line.

ovs−vswitchd switches may be configured with any of the following features:

- L2 switching with MAC learning.
- NIC bonding with automatic fail-over and source MAC-based TX load balancing ("SLB").
- 802.1Q VLAN support.
- Port mirroring, with optional VLAN tagging.
- NetFlow v5 flow logging.
- sFlow(R) monitoring.
- Connectivity to an external OpenFlow controller, such as NOX.

Only a single instance of **ovs−vswitchd** is intended to run at a time. A single **ovs−vswitchd** can manage any number of switch instances, up to the maximum number of supported Open vSwitch datapaths.

ovs−vswitchd does all the necessary management of Open vSwitch datapaths itself. Thus, external tools, such **ovs−dpctl**(8), are not needed for managing datapaths in conjunction with **ovs−vswitchd**, and their use to modify datapaths when **ovs−vswitchd** is running can interfere with its operation. (**ovs−dpctl** may still be useful for diagnostics.)

An Open vSwitch datapath kernel module must be loaded for **ovs−vswitchd** to be useful. Please refer to the **INSTALL.Linux** file included in the Open vSwitch distribution for instructions on how to build and

load the Open vSwitch kernel module.

OPTIONS

−−mlockall

Causes **ovs−vswitchd** to call the **mlockall()** function, to attempt to lock all of its process memory into physical RAM, preventing the kernel from paging any of its memory to disk. This helps to avoid networking interruptions due to system memory pressure.

Some systems do not support **mlockall()** at all, and other systems only allow privileged users, such as the superuser, to use it. **ovs−vswitchd** emits a log message if **mlockall()** is unavailable or unsuccessful.

−−pidfile[=*pidfile*]

Causes a file (by default, **ovs−vswitchd.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**.

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−vswitchd** 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

Causes **ovs−vswitchd** to detach itself from the foreground session and run as a background process. **ovs−vswitchd** detaches only after it has connected to the database, retrieved the initial configuration, and set up that configuration.

−−monitor

Creates an additional process to monitor the **ovs−vswitchd** daemon. If the daemon dies due to a signal that indicates a programming error (e.g. **SIGSEGV**, **SIGABRT**), then the monitor process starts a new copy of it. If the daemon die 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−vswitchd** changes its current working directory to the root directory after it detaches. Otherwise, invoking **ovs−vswitchd** 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−vswitchd** 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.

Public Key Infrastructure Options

−p *privkey.pem*
−−private−key=*privkey.pem*

Specifies a PEM file containing the private key used as **ovs−vswitchd**'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−vswitchd** should use to verify certificates presented to it by SSL peers. (This may be the same certificate that SSL peers use to verify the certificate specified on **−c** or **−−certificate**, or it may be a different one, depending on the PKI design in use.)

−C none
−−ca−cert=none
Disables verification of certificates presented by SSL peers. This introduces a security risk, because it means that certificates cannot be verified to be those of known trusted hosts.

−−bootstrap−ca−cert=*cacert.pem*
When *cacert.pem* exists, this option has the same effect as **−C** or **−−ca−cert**. If it does not exist, then **ovs−vswitchd** will attempt to obtain the CA certificate from the SSL peer on its first SSL connection and save it to the named PEM file. If it is successful, it will immediately drop the connection and reconnect, and from then on all SSL connections must be authenticated by a certificate signed by the CA certificate thus obtained.

This option exposes the SSL connection to a man-in-the-middle attack obtaining the initial CA certificate, but it may be useful for bootstrapping.

This option is only useful if the SSL peer sends its CA certificate as part of the SSL certificate chain. The SSL protocol does not require the server to send the CA certificate, but **ovs−controller**(8) can be configured to do so with the **−−peer−ca−cert** option.

This option is mutually exclusive with **−C** and **−−ca−cert**.

−v[*spec*]
−−verbose=[*spec*]
Sets logging levels. Without any *spec*, sets the log level for every module and facility 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.
- **off**, **emer**, **err**, **warn**, **info**, or **dbg**, to control the log level. Messages of the given severity or higher will be logged, and messages of lower severity will be filtered out. **off** filters out all messages. See **ovs−appctl**(8) for a definition of each log level.

Case is not significant within *spec*.

Regardless of the log levels set for **file**, logging to a file will not take place unless **−−log−file** is also specified (see below).

For compatibility with older versions of OVS, **any** is accepted as a word but has no effect.

−v
−−verbose
Sets the maximum logging verbosity level, equivalent to **−−verbose=dbg**.

−−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−vswitchd.log**.

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

−V

−−version
Prints version information to the console.

−−check−leaks=*file*
Logs information about memory allocation and deallocation to *file*, to allow for debugging memory leaks in **ovs−vswitchd**. This option slows down **ovs−vswitchd** considerably, so it should only be used when a memory leak is suspected. Use the **ovs−parse−leaks** script to interpret the leak file.

−−leak−limit=*size*
Limits size of the leak file as specified by **−−check−leaks** to *size* bytes. Finding leaks sometimes requires allowing the leak file to grow very large, up to 1GB. By default, files are limited to 10MB.

RUNTIME MANAGEMENT COMMANDS

ovs−appctl(8) can send commands to a running **ovs−vswitchd** process. The currently supported commands are described below. The command descriptions assume an understanding of how to configure Open vSwitch.

GENERAL COMMANDS

exit Causes **ovs−vswitchd** to gracefully terminate.

qos/show *interface*
Queries the kernel for Quality of Service configuration and statistics associated with the given *interface*.

cfm/show [*interface*]
Displays detailed information about Connectivity Fault Management configured on *interface*. If *interface* is not specified, then displays detailed information about all interfaces with CFM enabled.

cfm/set-fault [*interface*] *status*
Force the fault status of the CFM module on *interface* (or all interfaces if none is given) to be *status*. *status* can be "true", "false", or "normal" which reverts to the standard behavior.

stp/tcn [*bridge*]
Forces a topology change event on *bridge* if it's running STP. This may cause it to send Topology Change Notifications to its peers and flush its MAC table.. If no *bridge* is given, forces a topology change event on all bridges.

BRIDGE COMMANDS

These commands manage bridges.

fdb/flush [*bridge*]
Flushes *bridge* MAC address learning table, or all learning tables if no *bridge* is given.

fdb/show *bridge*
Lists each MAC address/VLAN pair learned by the specified *bridge*, along with the port on which it was learned and the age of the entry, in seconds.

bridge/reconnect [*bridge*]
Makes *bridge* drop all of its OpenFlow controller connections and reconnect. If *bridge* is not specified, then all bridges drop their controller connections and reconnect.

This command might be useful for debugging OpenFlow controller issues.

bridge/dump−flows *bridge*
Lists all flows in *bridge*, including those normally hidden to commands such as **ovs−ofctl dump−flows**. Flows set up by mechanisms such as in-band control and fail-open are hidden from the controller since it is not allowed to modify or override them.

BOND COMMANDS

These commands manage bonded ports on an Open vSwitch's bridges. To understand some of these commands, it is important to understand a detail of the bonding implementation called ''source load balancing'' (SLB). Instead of directly assigning Ethernet source addresses to slaves, the bonding implementation computes a function that maps an 48-bit Ethernet source addresses into an 8-bit value (a ''MAC hash'' value). All of the Ethernet addresses that map to a single 8-bit value are then assigned to a single slave.

bond/list

Lists all of the bonds, and their slaves, on each bridge.

bond/show [*port*]

Lists all of the bond-specific information (updelay, downdelay, time until the next rebalance) about the given bonded *port*, or all bonded ports if no *port* is given. Also lists information about each slave: whether it is enabled or disabled, the time to completion of an updelay or downdelay if one is in progress, whether it is the active slave, the hashes assigned to the slave. Any LACP information related to this bond may be found using the **lacp/show** command.

bond/migrate *port hash slave*

Only valid for SLB bonds. Assigns a given MAC hash to a new slave. *port* specifies the bond port, *hash* the MAC hash to be migrated (as a decimal number between 0 and 255), and *slave* the new slave to be assigned.

The reassignment is not permanent: rebalancing or fail-over will cause the MAC hash to be shifted to a new slave in the usual manner.

A MAC hash cannot be migrated to a disabled slave.

bond/set−active−slave *port slave*

Sets *slave* as the active slave on *port*. *slave* must currently be enabled.

The setting is not permanent: a new active slave will be selected if *slave* becomes disabled.

bond/enable−slave *port slave*
bond/disable−slave *port slave*

Enables (or disables) *slave* on the given bond *port*, skipping any updelay (or downdelay).

This setting is not permanent: it persists only until the carrier status of *slave* changes.

bond/hash *mac* [*vlan*] [*basis*]

Returns the hash value which would be used for *mac* with *vlan* and *basis* if specified.

lacp/show [*port*]

Lists all of the LACP related information about the given *port*: active or passive, aggregation key, system id, and system priority. Also lists information about each slave: whether it is enabled or disabled, whether it is attached or detached, port id and priority, actor information, and partner information. If *port* is not specified, then displays detailed information about all interfaces with CFM enabled.

DATAPATH COMMANDS

These commands manage logical datapaths. They are are similar to the equivalent **ovs−dpctl** commands.

dpif/dump−dps

Prints the name of each configured datapath on a separate line.

dpif/show [*dp*...]

Prints a summary of configured datapaths, including statistics and a list of connected ports. The port information includes the OpenFlow port number, datapath port number, and the type. (The local port is identified as OpenFlow port 65534.)

If one or more datapaths are specified, information on only those datapaths are displayed. Otherwise, information about all configured datapaths are shown.

dpif/dump−flows *dp*

Prints to the console all flow entries in datapath *dp*'s flow table.

This command is primarily useful for debugging Open vSwitch. The flow table entries that it displays are not OpenFlow flow entries. Instead, they are different and considerably simpler flows maintained by the datapath module. If you wish to see the OpenFlow flow entries, use **ovs−ofctl dump−flows**.

dpif/del−flows *dp*

Deletes all flow entries from datapath *dp*'s flow table and underlying datapath implementation (e.g., kernel datapath module).

This command is primarily useful for debugging Open vSwitch. As discussed in **dpif/dump−flows**, these entries are not OpenFlow flow entries.

OFPROTO COMMANDS

These commands manage the core OpenFlow switch implementation (called **ofproto**).

ofproto/list

Lists the names of the running ofproto instances. These are the names that may be used on **ofproto/trace**.

ofproto/trace *switch priority tun_id in_port mark packet*
ofproto/trace *switch flow* **−generate**

Traces the path of an imaginary packet through *switch*. Both forms require *switch*, the switch on which the packet arrived (one of those listed by **ofproto/list**). The first form specifies a packet's contents explicitly:

priority Packet QoS priority. Use **0** if QoS is not setup.

tun_id The tunnel ID on which the packet arrived. Use **0** if the packet did not arrive through a tunnel.

in_port The OpenFlow port on which the packet arrived. Use **65534** if the packet arrived on **OFPP_LOCAL**, the local port.

mark SKB mark of the packet. Use **0** if Netfilter marks are not used.

packet A sequence of hex digits specifying the packet's contents. An Ethernet frame is at least 14 bytes long, so there must be at least 28 hex digits. Obviously, it is inconvenient to type in the hex digits by hand, so the **ovs−pcap**(1) and **ovs−tcpundump**(1) utilities provide easier ways.

The second form specifies the packet's contents implicitly:

flow A flow in one of two forms: either the form printed by **ovs−dpctl**(8)'s **dump−flows** command, or in a format 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. **ovs−vswitchd** generates an arbitrary packet that has the specified *flow*.

ovs−vswitchd will respond with extensive information on how the packet would be handled if it were to be received. The packet will not actually be sent, but side effects such as MAC learning will occur.

ofproto/trace *switch flow*

Traces the path of a packet in an imaginary flow through *switch*. The arguments are:

switch The switch on which the packet arrived (one of those listed by **ofproto/list**).

flow A flow in one of two forms: either the form printed by **ovs−dpctl**(8)'s **dump−flows** command, or in a format 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.

ovs−vswitchd will respond with extensive information on how a packet in *flow* would be handled if it were received by *switch*. No packet will actually be sent. Some side effects may occur, but

MAC learning in particular will not.

This form of **ofproto/trace** cannot determine the complete set of datapath actions in some corner cases. If the results say that this is the case, rerun **ofproto/trace** supplying a packet in the flow to get complete results.

ofproto/self−check [*switch*]
Runs an internal consistency check on *switch*, if specified, otherwise on all ofproto instances, and responds with a brief summary of the results. If the summary reports any errors, then the Open vSwitch logs should contain more detailed information. Please pass along errors reported by this command to the Open vSwitch developers as bugs.

VLOG COMMANDS

These commands manage **ovs−vswitchd**'s logging settings.

vlog/set [*spec*]
Sets logging levels. Without any *spec*, sets the log level for every module and facility 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.
- **off**, **emer**, **err**, **warn**, **info**, or **dbg**, to control the log level. Messages of the given severity or higher will be logged, and messages of lower severity will be filtered out. **off** filters out all messages. See **ovs−appctl**(8) for a definition of each log level.

Case is not significant within *spec*.

Regardless of the log levels set for **file**, logging to a file will not take place unless **ovs−vswitchd** was invoked with the **−−log−file** option.

For compatibility with older versions of OVS, **any** is accepted as a word but has no effect.

vlog/set PATTERN:*facility***:***pattern*
Sets the log pattern for *facility* to *pattern*. Refer to **ovs−appctl**(8) for a description of the valid syntax for *pattern*.

vlog/list
Lists the supported logging modules and their current levels.

vlog/reopen
Causes **ovs−vswitchd** to close and reopen its log file. (This is useful after rotating log files, to cause a new log file to be used.)

This has no effect unless **ovs−vswitchd** was invoked with the **−−log−file** option.

vlog/disable−rate−limit [*module*]...
vlog/enable−rate−limit [*module*]...
By default, **ovs−vswitchd** limits the rate at which certain messages can be logged. When a message would appear more frequently than the limit, it is suppressed. This saves disk space, makes logs easier to read, and speeds up execution, but occasionally troubleshooting requires more detail. Therefore, **vlog/disable−rate−limit** allows rate limits to be disabled at the level of an individual log module. Specify one or more module names, as displayed by the **vlog/list** command. Specifying either no module names at all or the keyword **any** disables rate limits for every log module.

The **vlog/enable−rate−limit** command, whose syntax is the same as **vlog/disable−rate−limit**, can be used to re-enable a rate limit that was previously disabled.

MEMORY COMMANDS

These commands report memory usage.

memory/show
Displays some basic statistics about **ovs−vswitchd**'s memory usage. **ovs−vswitchd** also logs this information soon after startup and periodically as its memory consumption grows.

COVERAGE COMMANDS

These commands manage **ovs−vswitchd**'s ''coverage counters,'' which count the number of times particular events occur during a daemon's runtime. In addition to these commands, **ovs−vswitchd** automatically logs coverage counter values, at **INFO** level, when it detects that the daemon's main loop takes unusually long to run.

Coverage counters are useful mainly for performance analysis and debugging.

coverage/show
Displays the values of all of the coverage counters.

STRESS OPTION COMMANDS

These command manage stress options, which allow developers testing Open vSwitch to trigger behavior that otherwise would occur only in corner cases. Developers and testers can thereby more easily discover bugs that would otherwise manifest only rarely or nondeterministically. Stress options may cause surprising behavior even when they do not actually reveal bugs, so they should only be enabled as part of testing Open vSwitch.

stress/enable
stress/disable
All stress options are disabled by default. Use **stress/enable** to enable stress options and **stress/disable** to disable them.

stress/list
Lists and describes the available stress options and their settings in tabular form. The columns in the table are:

NAME A single-word identifier for the option, used to identify stress options to **stress/set**.

DESCRIPTION
A description for a person unfamiliar with the detailed internals of the code what behavior the option affects.

PERIOD
Currently configured trigger period. If the stress option is disabled, this is **disabled**. Otherwise this is a number giving the number of occurrences of the event between activations of the stress option triggers.

MODE If the stress option is disabled, this is **n/a**. Otherwise it is **periodic** if the stress option triggers after exactly the period, or **random** if it triggers randomly but on average after the number of occurrences specified by the period.

COUNTER
If the stress option is disabled, this is **n/a**. Otherwise it is the number of occurrences of the event before the next time the stress option triggers.

HITS The number of times that this stress option has triggered since this program started.

RECOMMENDED
A suggested period for a person unfamiliar with the internals. It should put reasonable stress on the system without crippling it.

MINIMUM
MAXIMUM
Minimum and maximum values allowed for the period.

DEFAULT
The default period, used when stress options have been enabled (with **stress/enable**) but this particular stress option has not been specifically configured (with **stress/set**). It is

disabled if the option is disabled by default. It is nonzero for options that can be left on at low levels without noticeable impact to the end user.

stress/set *option period* [**random**|**periodic**]

Sets the period at which stress *option* triggers to *period*. A *period* of 0 disables *option*. Specify **random** to make the option trigger randomly with an average period of *period*, or **periodic** to trigger exactly every *period* events; the latter is the default.

If stress options have not been enabled with **stress/enable**, this command has no effect.

LIMITS

We believe these limits to be accurate as of this writing. These limits assume the use of the Linux kernel datapath.

- Approximately 256 bridges given the allowance of 5,000 file descriptors that **ovs−ctl**(8) configures. (**ovs−vswitchd** requires 17 file descriptors per datapath.)
- 65,280 ports per bridge. Performance will degrade beyond 1,024 ports per bridge due to fixed hash table sizing.
- 2,048 MAC learning entries per bridge.
- Kernel flows are limited only by memory available to the kernel. Performance will degrade beyond 1,048,576 kernel flows per bridge with a 32-bit kernel, beyond 262,144 with a 64-bit kernel. (**ovs−vswitchd** should never install anywhere near that many flows.)
- OpenFlow flows are limited only by available memory. Performance is linear in the number of unique wildcard patterns. That is, an OpenFlow table that contains many flows that all match on the same fields in the same way has a constant-time lookup, but a table that contains many flows that match on different fields requires lookup time linear in the number of flows.
- 255 ports per bridge participating in 802.1D Spanning Tree Protocol.
- 32 mirrors per bridge.
- 15 bytes for the name of a port. (This is a Linux kernel limitation.)

SEE ALSO

ovs−appctl(8), **ovsdb−server**(1), **INSTALL.Linux** in the Open vSwitch distribution.