NAME

ovn-controller – Open Virtual Network local controller

SYNOPSIS

ovn−controller [*options*] [*ovs-database*]

DESCRIPTION

ovn−controller is the local controller daemon for OVN, the Open Virtual Network. It connects up to the OVN Southbound database (see **ovn−sb**(5)) over the OVSDB protocol, and down to the Open vSwitch database (see **ovs−vswitchd.conf.db**(5)) over the OVSDB protocol and to **ovs−vswitchd**(8) via OpenFlow. Each hypervisor and software gateway in an OVN deployment runs its own independent copy of **ovn−controller**; thus, **ovn−controller**'s downward connections are machine-local and do not run over a physical network.

ACL LOGGING

ACL log messages are logged through **ovn−controller**'s logging mechanism. ACL log entries have the module **acl_log** at log level **info**. Configuring logging is described below in the **Logging Options** section.

OPTIONS

Daemon Options

−−pidfile[**=***pidfile*]

Causes a file (by default, *program***.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 .

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, the daemon 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 this program as a background process. The process forks, and in the child it starts a new session, closes the standard file descriptors (which has the side effect of disabling logging to the console), and changes its current directory to the root (unless **−−no−chdir** is specified). After the child completes its initialization, the parent exits.

−−monitor

Creates an additional process to monitor this program. If it 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, the daemon changes its current working directory to the root directory after it detaches. Otherwise, invoking the daemon 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 the daemon 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 this daemon will try to self-confine itself to work with files under well-known directories determined at build time. 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=*user***:***group*
Causes this program 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 assumed, respectively. Only daemons started by the root user accepts this argument.

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

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

Logging Options

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

- A valid module name, as displayed by the **vlog/list** command on **ovs−appctl**(8), limits the log level change to the specified module.
- **syslog**, **console**, or **file**, to limit the log level change to only to the system log, to the console, or to a file, respectively. (If **−−detach** is specified, the daemon closes its standard file descriptors, so logging to the console will have no effect.)

  On Windows platform, **syslog** is accepted as a word and is only useful along with the **−−syslog−target** option (the word has no effect otherwise).
- **off**, **emer**, **err**, **warn**, **info**, or **dbg**, to control the log level. Messages of the given severity or higher will be logged, and messages of lower severity will be filtered out. **off** filters out all messages. See **ovs−appctl**(8) for a definition of each log level.

Case is not significant within *spec*.

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

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

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

−vPATTERN:*destination***:***pattern*
−−verbose=PATTERN:*destination***:***pattern*
Sets the log pattern for *destination* to *pattern*. Refer to **ovs−appctl**(8) for a description of the valid syntax for *pattern*.

−vFACILITY:*facility*
−−verbose=FACILITY:*facility*
Sets the RFC5424 facility of the log message. *facility* can be one of **kern**, **user**, **mail**, **daemon**, **auth**, **syslog**, **lpr**, **news**, **uucp**, **clock**, **ftp**, **ntp**, **audit**, **alert**, **clock2**, **local0**, **local1**, **local2**, **local3**, **local4**, **local5**, **local6** or **local7**. If this option is not specified, **daemon** is used as the default for the local system syslog and **local0** is used while sending a message to the target provided via the **−−syslog−target** option.

−−log−file[**=***file*]
Enables logging to a file. If *file* is specified, then it is used as the exact name for the log file. The default log file name used if *file* is omitted is **/usr/local/var/log/ovn/***program***.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* as how syslog messages should be sent to syslog daemon. The following forms are supported:

- **libc**, to use the 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*, to use a 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*, to use a 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**, to discard all messages logged to syslog.

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

PKI Options
PKI configuration is required in order to use SSL for the connections to the Northbound and Southbound databases.

−p *privkey.pem*
−−private−key=*privkey.pem*
Specifies a PEM file containing the private key used as 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 for verifying certificates presented to this program 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 the executable 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.

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

−−peer−ca−cert=*peer-cacert.pem*

Specifies a PEM file that contains one or more additional certificates to send to SSL peers. *peer-cacert.pem* should be the CA certificate used to sign the program's own certificate, that is, the certificate specified on **−c** or **−−certificate**. If the program's certificate is self-signed, then **−−certificate** and **−−peer−ca−cert** should specify the same file.

This option is not useful in normal operation, because the SSL peer must already have the CA certificate for the peer to have any confidence in the program's identity. However, this offers a way for a new installation to bootstrap the CA certificate on its first SSL connection.

Other Options

−−unixctl=*socket*

Sets the name of the control socket on which *program* listens for runtime management commands (see *RUNTIME MANAGEMENT COMMANDS,* below). If *socket* does not begin with **/**, it is interpreted as relative to . If **−−unixctl** is not used at all, the default socket is **/***program***.***pid***.ctl**, where *pid* is *program*'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 *program* 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.

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

−V
−−version

Prints version information to the console.

CONFIGURATION

ovn−controller retrieves most of its configuration information from the local Open vSwitch's ovsdb-server instance. The default location is **db.sock** in the local Open vSwitch's "run" directory. It may be overridden by specifying the *ovs-database* argument as an OVSDB active or passive connection method, as described in **ovsdb**(7).

ovn−controller assumes it gets configuration information from the following keys in the **Open_vSwitch** table of the local OVS instance:

external_ids:system−id

The chassis name to use in the Chassis table. Changing the **system−id** while **ovn−controller** is running is not directly supported. Users have two options: either first gracefully stop **ovn−controller** or manually delete the stale **Chassis** and **Chassis_Private** records after changing the **system−id**. Note that the chassis name can also be provided via the

system−id−override file in the local OVN "etc" directory or via the **−n** command-line option. The following precedence is used: first, the command-line option is read; if not present, the **system−id−override** file is read; if not present, then the name configured in the database is used.

external_ids:hostname
The hostname to use in the Chassis table.

external_ids:ovn−bridge
The integration bridge to which logical ports are attached. The default is **br−int**. If this bridge does not exist when ovn-controller starts, it will be created automatically with the default configuration suggested in **ovn−architecture**(7). When more than one controllers are running on the same host, **external_ids:ovn−bridge−CHASSIS_NAME** should be set for each of them, pointing to a unique bridge. This is required to avoid controllers stepping on each others' feet.

external_ids:ovn−bridge−datapath−type
This configuration is optional. If set, then the datapath type of the integration bridge will be set to the configured value. If this option is not set, then **ovn−controller** will not modify the existing **datapath−type** of the integration bridge.

external_ids:ovn−remote
The OVN database that this system should connect to for its configuration, in one of the same forms documented above for the *ovs-database*.

external_ids:ovn−monitor−all
A boolean value that tells if **ovn−controller** should monitor all records of tables in *ovs-database*. If set to **false**, it will conditionally monitor the records that is needed in the current chassis.

It is more efficient to set it to **true** in use cases where the chassis would anyway need to monitor most of the records in *OVN Southbound* database, which would save the overhead of conditions processing, especially for server side. Typically, set it to **true** for environments that all workloads need to be reachable from each other.

NOTE: for efficiency and scalability in common scenarios **ovn−controller** unconditionally monitors all sub-ports (ports with **parent_port** set) regardless of the **ovn−monitor−all** value.

Default value is *false*.

external_ids:ovn−remote−probe−interval
The inactivity probe interval of the connection to the OVN database, in milliseconds. If the value is zero, it disables the connection keepalive feature.

If the value is nonzero, then it will be forced to a value of at least 1000 ms.

external_ids:ovn−encap−type
The encapsulation type that a chassis should use to connect to this node. Multiple encapsulation types may be specified with a comma-separated list. Each listed encapsulation type will be paired with **ovn−encap−ip**.

Supported tunnel types for connecting hypervisors and gateways are **geneve**, **vxlan**, and **stt**.

Due to the limited amount of metadata in **vxlan**, the capabilities and performance of connected gateways and hypervisors will be reduced versus other tunnel formats.

external_ids:ovn−encap−ip
The IP address that a chassis should use to connect to this node using encapsulation types specified by **external_ids:ovn−encap−type**. Multiple encapsulation IPs may be specified with a comma-separated list.

In scenarios where multiple encapsulation IPs are present, distinct tunnels are established for each remote chassis. These tunnels are differentiated by setting unique **options:local_ip** and **options:remote_ip** values in the tunnel interface. When transmitting a packet to a remote chassis, the selection of local_ip is guided by the **Interface:external_ids:encap−ip** from the local OVSDB, corresponding to the VIF originating the packet, if specified. The **Interface:external_ids:encap−ip** setting of the VIF is also populated to the **Port_Binding** table in the OVN SB database via the **encap** column. Consequently, when a remote chassis needs to send a packet to a port-binding associated with this VIF, it utilizes the tunnel with the appropriate **options:remote_ip** that matches the **ip** in **Port_Binding:encap**. This mechanism is particularly beneficial for chassis with multiple physical interfaces designated for tunneling, where each interface is optimized for handling specific traffic associated with particular VIFs.

external_ids:ovn−encap−ip−default
When **ovn−encap−ip** contains multiple IPs, this field indicates the default one.

external_ids:ovn−encap−df_default
indicates the DF flag handling of the encapulation. Set to **true** to set the DF flag for new data paths or **false** to clear the DF flag.

external_ids:ovn−bridge−mappings
A list of key-value pairs that map a physical network name to a local ovs bridge that provides connectivity to that network. An example value mapping two physical network names to two ovs bridges would be: **physnet1:br−eth0,physnet2:br−eth1**.

external_ids:ovn−encap−csum
ovn−encap−csum indicates that encapsulation checksums can be transmitted and received with reasonable performance. It is a hint to senders transmitting data to this chassis that they should use checksums to protect OVN metadata. Set to **true** to enable or **false** to disable. Depending on the capabilities of the network interface card, enabling encapsulation checksum may incur performance loss. In such cases, encapsulation checksums can be disabled.

external_ids:ovn−encap−tos
ovn−encap−tos indicates the value to be applied to OVN tunnel interface's option:tos as specified in the Open_vSwitch database Interface table. Please refer to Open VSwitch Manual for details.

external_ids:ovn−cms−options
A list of options that will be consumed by the CMS Plugin and which specific to this particular chassis. An example would be: **cms_option1,cms_option2:foo**.

external_ids:ovn−transport−zones
The transport zone(s) that this chassis belongs to. Transport zones is a way to group different chassis so that tunnels are only formed between members of the same group(s). Multiple transport zones may be specified with a comma-separated list. For example: tz1,tz2,tz3.

If not set, the Chassis will be considered part of a default transport zone.

external_ids:ovn−chassis−mac−mappings
A list of key-value pairs that map a chassis specific mac to a physical network name. An example value mapping two chassis macs to two physical network names would be: **physnet1:aa:bb:cc:dd:ee:ff,physnet2:a1:b2:c3:d4:e5:f6**. These are the macs that ovn-controller will replace a router port mac with, if packet is going from a distributed router port on vlan type logical switch.

external_ids:ovn−is−interconn
The boolean flag indicates if the chassis is used as an interconnection gateway.

external_ids:ovn−match−northd−version
The boolean flag indicates if **ovn−controller** needs to check **ovn−northd** version. If this flag is set to true and the **ovn−northd's** version (reported in the Southbound database) doesn't match with the **ovn−controller's** internal version, then it will stop processing the southbound and local Open vSwitch database changes. The default value is considered false if this option is not defined.

external_ids:ovn−ofctrl−wait−before−clear
The time, in milliseconds, to wait before clearing flows in OVS after OpenFlow connection/reconnection during **ovn−controller** initialization. The purpose of this wait is to give time for **ovn−controller** to compute the new flows before clearing existing ones, to avoid data plane down time during **ovn−controller** restart/upgrade at large scale environments where recomputing the flows takes more than a few seconds or even longer. It is difficult for **ovn−controller** to determine when the new flows computing is completed, because of the dynamics in the cloud environments, which is why this configuration is provided for users to adjust based on the scale of the environment. By default, it is 0, which means clearing existing flows without waiting. Not setting the value, or setting it too small, may result in data plane down time during upgrade/restart, while setting it too big may result in unnecessary extra control plane latency of applying new changes of CMS during upgrade/restart. In most cases, a slightly bigger value is not harmful, because the extra control plane latency happens only once during the OpenFlow connection. To get a reasonable range of the value setting, it is recommended to run the below commands on a node in the target environment and then set this configuration to twice the value of **Maximum** shown in the output of the second command.

- **ovn−appctl −t ovn−controller inc−engine/recompute**
- **ovn−appctl −t ovn−controller stopwatch/show flow−generation**

external_ids:ovn−enable−lflow−cache
The boolean flag indicates if **ovn−controller** should enable/disable the logical flow in-memory cache it uses when processing Southbound database logical flow changes. By default caching is enabled.

external_ids:ovn−limit−lflow−cache
When used, this configuration value determines the maximum number of logical flow cache entries **ovn−controller** may create when the logical flow cache is enabled. By default the size of the cache is unlimited.

external_ids:ovn−memlimit−lflow−cache−kb
When used, this configuration value determines the maximum size of the logical flow cache (in KB) **ovn−controller** may create when the logical flow cache is enabled. By default the size of the cache is unlimited.

external_ids:ovn−trim−limit−lflow−cache
When used, this configuration value sets the minimum number of entries in the logical flow cache starting with which automatic memory trimming is performed. By default this is set to 10000 entries.

external_ids:ovn−trim−wmark−perc−lflow−cache
When used, this configuration value sets the percentage from the high watermark number of entries in the logical flow cache under which automatic memory trimming is performed. E.g., if the trim watermark percentage is set to 50%, automatic memory trimming happens only when the number of entries in the logical flow cache gets reduced to less than half of the last measured high watermark. By default this is set to 50.

external_ids:ovn−trim−timeout−ms
When used, this configuration value specifies the time, in milliseconds, since the last logical flow cache operation after which **ovn−controller** performs memory trimming regardless of how many entries there are in the cache. By default this is set to 30000 (30

seconds).

external_ids:garp−max−timeout−sec
When used, this configuration value specifies the maximum timeout (in seconds) between two consecutive GARP packets sent by **ovn−controller**. **ovn−controller** by default sends just 4 GARP packets with an exponential backoff timeout. Setting **external_ids:garp−max−timeout−sec** allows to cap for the exponential backoff used by **ovn−controller** to send GARPs packets.

external_ids:ovn−bridge−remote
Connection to the OVN management bridge in OvS. It defaults to **unix:***br-int***.mgmt** when not specified.

external_ids:ovn−bridge−remote−probe−interval
The inactivity probe interval of the connection to the OVN management bridge, in milliseconds. It defaults to zero. If the value is zero, it disables the inactivity probe.

Most of configuration options listed above can also be set for a particular chassis name (see **external_ids:system−id** for more information). This can be achieved by setting **external_ids:option−[chassis]** instead of **external_ids:option**. For example, set **external_ids:ovn−encap−ip−otherhv** to use a particular IP address for the controller instance named **otherhv**. Name specific configuration options always override any global options set in the database.

Chassis-specific configuration options in the database plus the ability to configure the chassis name to use via the **system−id−override** file or command line allows to run multiple **ovn−controller** instances with unique chassis names on the same host using the same **vswitchd** instance. This may be useful when running a hybrid setup with more than one CMS managing ports on the host, or to use different datapath types on the same host. Also note that this ability is highly experimental and has known limitations (for example, stateful ACLs are not supported). Use at your own risk.

ovn−controller reads the following values from the **Open_vSwitch** database of the local OVS instance:

datapath−type from **Bridge** table
This value is read from local OVS integration bridge row of **Bridge** table and populated in **other_config:datapath-type** of the **Chassis** table in the OVN_Southbound database.

iface−types from **Open_vSwitch** table
This value is populated in **external_ids:iface-types** of the **Chassis** table in the OVN_Southbound database.

private_key, **certificate**, **ca_cert**, and **bootstrap_ca_cert** from **SSL** table
These values provide the SSL configuration used for connecting to the OVN southbound database server when an SSL connection type is configured via **external_ids:ovn−remote**. Note that this SSL configuration can also be provided via command-line options, the configuration in the database takes precedence if both are present.

OPEN VSWITCH DATABASE USAGE

ovn−controller uses a number of **external_ids** keys in the Open vSwitch database to keep track of ports and interfaces. For proper operation, users should not change or clear these keys:

external_ids:ovn−chassis−id in the **Port** table
The presence of this key identifies a tunnel port within the integration bridge as one created by **ovn−controller** to reach a remote chassis. Its value is the chassis ID of the remote chassis.

external_ids:ct−zone−range in the **Open_vSwitch** table
The presence of this key identifies a minimum and maximum values for ct-zone ids dynamically selected by ovn-controller (boundaries are included in the range). Minimum value is 1 while maximum value is 65535.

external_ids:ct−zone−* in the **Bridge** table
: Logical ports and gateway routers are assigned a connection tracking zone by **ovn−controller** for stateful services. To keep state across restarts of **ovn−controller**, these keys are stored in the integration bridge's Bridge table. The name contains a prefix of **ct−zone−** followed by the name of the logical port or gateway router's zone key. The value for this key identifies the zone used for this port.

external_ids:ovn−localnet−port in the **Port** table
: The presence of this key identifies a patch port as one created by **ovn−controller** to connect the integration bridge and another bridge to implement a **localnet** logical port. Its value is the name of the logical port with **type** set to **localnet** that the port implements. See **external_ids:ovn−bridge−mappings**, above, for more information.

  Each **localnet** logical port is implemented as a pair of patch ports, one in the integration bridge, one in a different bridge, with the same **external_ids:ovn−localnet−port** value.

external_ids:ovn−l2gateway−port in the **Port** table
: The presence of this key identifies a patch port as one created by **ovn−controller** to connect the integration bridge and another bridge to implement a **l2gateway** logical port. Its value is the name of the logical port with **type** set to **l2gateway** that the port implements. See **external_ids:ovn−bridge−mappings**, above, for more information.

  Each **l2gateway** logical port is implemented as a pair of patch ports, one in the integration bridge, one in a different bridge, with the same **external_ids:ovn−l2gateway−port** value.

external−ids:ovn−l3gateway−port in the **Port** table
: This key identifies a patch port as one created by **ovn−controller** to implement a **l3gateway** logical port. Its value is the name of the logical port with type set to **l3gateway**. This patch port is similar to the OVN logical patch port, except that **l3gateway** port can only be bound to a particular chassis.

external−ids:ovn−logical−patch−port in the **Port** table
: This key identifies a patch port as one created by **ovn−controller** to implement an OVN logical patch port within the integration bridge. Its value is the name of the OVN logical patch port that it implements.

external−ids:ovn−startup−ts in the **Bridge** table
: This key represents the timestamp (in milliseconds) at which **ovn−controller** process was started.

external−ids:ovn−nb−cfg in the **Bridge** table
: This key represents the last known **OVN_Southbound.SB_Global.nb_cfg** value for which all flows have been successfully installed in OVS.

external−ids:ovn−nb−cfg−ts in the **Bridge** table
: This key represents the timestamp (in milliseconds) of the last known **OVN_Southbound.SB_Global.nb_cfg** value for which all flows have been successfully installed in OVS.

external_ids:ovn−installed and **external_ids:ovn−installed−ts** in the **Interface** table
: This key is set after all openflow operations corresponding to the OVS interface have been processed by ovs-vswitchd. At the same time a timestamp, in milliseconds since the epoch, is stored in **external_ids:ovn−installed−ts**.

OVN SOUTHBOUND DATABASE USAGE

ovn−controller reads from much of the **OVN_Southbound** database to guide its operation. **ovn−controller** also writes to the following tables:

Chassis
: Upon startup, **ovn−controller** creates a row in this table to represent its own chassis. Upon graceful termination, e.g. with **ovs−appctl −t ovn−controller exit** (but not

SIGTERM), **ovn−controller** removes its row.

Encap Upon startup, **ovn−controller** creates a row or rows in this table that represent the tunnel encapsulations by which its chassis can be reached, and points its **Chassis** row to them. Upon graceful termination, **ovn−controller** removes these rows.

Port_Binding
At runtime, **ovn−controller** sets the **chassis** columns of ports that are resident on its chassis to point to its **Chassis** row, and, conversely, clears the **chassis** column of ports that point to its **Chassis** row but are no longer resident on its chassis. The **chassis** column has a weak reference type, so when **ovn−controller** gracefully exits and removes its **Chassis** row, the database server automatically clears any remaining references to that row.

MAC_Binding
At runtime, **ovn−controller** updates the **MAC_Binding** table as instructed by **put_arp** and **put_nd** logical actions. These changes persist beyond the lifetime of **ovn−controller**.

RUNTIME MANAGEMENT COMMANDS

ovs−appctl can send commands to a running **ovn−controller** process. The currently supported commands are described below.

exit Causes **ovn−controller** to gracefully terminate.

ct−zone−list
Lists each local logical port and its connection tracking zone.

meter−table−list
Lists each meter table entry and its local meter id.

group−table−list
Lists each group table entry and its local group id.

inject−pkt *microflow*
Injects *microflow* into the connected Open vSwitch instance. *microflow* must contain an ingress logical port (**inport** argument) that is present on the Open vSwitch instance.

The *microflow* argument describes the packet whose forwarding is to be simulated, in the syntax of an OVN logical expression, as described in **ovn−sb**(5), to express constraints. The parser understands prerequisites; for example, if the expression refers to **ip4.src**, there is no need to explicitly state **ip4** or **eth.type == 0x800**.

connection−status
Show OVN SBDB connection status for the chassis.

recompute
Trigger a full compute iteration in **ovn−controller** based on the contents of the Southbound database and local OVS database.

This command is intended to use only in the event of a bug in the incremental processing engine in **ovn−controller** to avoid inconsistent states. It should therefore be used with care as full recomputes are cpu intensive.

sb−cluster−state−reset
Reset southbound database cluster status when databases are destroyed and rebuilt.

If all databases in a clustered southbound database are removed from disk, then the stored index of all databases will be reset to zero. This will cause ovn-controller to be unable to read or write to the southbound database, because it will always detect the data as stale. In such a case, run this command so that ovn-controller will reset its local index so that it can interact with the southbound database again.

debug/delay−nb−cfg−report *seconds*
This command is used to delay ovn-controller updating the **nb_cfg** back to **OVN_Southbound** database. This is useful when **ovn−nbctl −−wait=hv** is used to measure end-to-end latency in a large scale environment. See **ovn−nbctl**(8) for more details.

lflow−cache/flush
Flushes the **ovn−controller** logical flow cache.

lflow−cache/show−stats
Displays logical flow cache statistics: enabled/disabled, per cache type entry counts.

inc−engine/show−stats
Display **ovn−controller** engine counters. For each engine node the following counters have been added:

- **recompute**
- **compute**
- **cancel**

inc−engine/show−stats *engine_node_name counter_name*
Display the **ovn−controller** engine counter(s) for the specified *engine_node_name*. *counter_name* is optional and can be one of **recompute**, **compute** or **cancel**.

inc−engine/clear−stats
Reset **ovn−controller** engine counters.