NAME

ovsdb−server − Open vSwitch database server

SYNOPSIS

ovsdb−server [*database*]... [**−−remote=***remote*]... [**−−run=***command*]

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

Service options:
[**−−service**] [**−−service−monitor**]

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

Syncing options:
[**−−sync−from=***server*]

Public key infrastructure options:
[**−−private−key=***privkey.pem*]
[**−−certificate=***cert.pem*]
[**−−ca−cert=***cacert.pem*]
[**−−bootstrap−ca−cert=***cacert.pem*]
[**−−peer−ca−cert=***peer-cacert.pem*]

SSL connection options:
[**−−ssl−protocols=***protocols*]
[**−−ssl−ciphers=***ciphers*]

Runtime management options:
−−unixctl=*socket*

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

DESCRIPTION

The **ovsdb−server** program provides RPC interfaces to one or more Open vSwitch databases (OVSDBs). It supports JSON-RPC client connections over active or passive TCP/IP or Unix domain sockets.

Each OVSDB file may be specified on the command line as *database*. If none is specified, the default is **/usr/local/etc/openvswitch/conf.db**. The database files must already have been created and initialized using, for example, **ovsdb−tool create**.

ACTIVE and BACKUP

ovsdb−server runs either as a backup server, or as an active server. When **ovsdb−server** is running as a backup server, all transactions that can modify the database content, including the lock commands are rejected. Active server, on the other hand, accepts all ovsdb server transactions. When **ovsdb−server** role changes, all existing client connection are reset, requiring clients to reconnect to the server.

By default, **ovsdb−server** runs as an active server, except when the **−−sync−from=***server* command line option is specified and without the **−−no−sync option**. During runtime, **ovsdb−server** role can be switch by using appctl commands.

ovsdb-server/connect−active−ovsdb−server switches **ovsdb−server** into a backup server, Conversely, **ovsdb-server/disconnect−active−ovsdb−server** switches server into an active one.

OPTIONS

−−remote=*remote*

Adds *remote* as a connection method used by **ovsdb−server**. *remote* must take one of the following forms:

pssl:*port*[**:***ip*]
Listen on the given SSL *port* for a connection. By default, connections are not bound to a particular local IP address and it listens only on IPv4 (but not IPv6) addresses, but specifying *ip* limits connections to those from the given *ip*, either IPv4 or IPv6 address. If *ip* is an IPv6 address, then wrap *ip* with square brackets, e.g.: **pssl:6640:[::1]**. 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 and it listens only on IPv4 (but not IPv6) addresses, but *ip* may be specified to listen only for connections to the given *ip*, either IPv4 or IPv6 address. If *ip* is an IPv6 address, then wrap *ip* with square brackets, e.g.: **ptcp:6640:[::1]**.

punix:*file*
On POSIX, listen on the Unix domain server socket named *file* for a connection.

On Windows, listen on a local named pipe. A file is created in the path *file* to mimic the behavior of a Unix domain socket.

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) in IPv4 or IPv6 address format. If *ip* is an IPv6 address, then wrap *ip* with square brackets, e.g.: **ssl:[::1]:6640**. 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*, where *ip* can be IPv4 or IPv6 address. If *ip* is an IPv6 address, then wrap *ip* with square brackets, e.g.: **tcp:[::1]:6640**.

unix:*file*
On POSIX, connect to the 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.

db:*db***,***table***,***column*
Reads additional connection methods from *column* in all of the rows in *table* within *db*. As the contents of *column* changes, **ovsdb−server** also adds and drops connection methods accordingly.

If *column*'s type is string or set of strings, then the connection methods are taken directly from the column. The connection methods in the column must have one of the forms described above.

If *column*'s type is UUID or set of UUIDs and references a table, then each UUID is looked up in the referenced table to obtain a row. The following columns in the row, if present and of the correct type, configure a connection method. Any additional columns are ignored.

target (string)
Connection method, in one of the forms described above. This column is mandatory: if it is missing or empty then no connection method can be configured.

max_backoff (integer)
Maximum number of milliseconds to wait between connection attempts.

inactivity_probe (integer)
Maximum number of milliseconds of idle time on connection to client before sending an inactivity probe message.

read_only (boolean)
If true, only read-only transactions are allowed on this connection.

It is an error for *column* to have another type.

To connect or listen on multiple connection methods, use multiple −−**remote** options.

−−**run=***command*]
Ordinarily **ovsdb−server** runs forever, or until it is told to exit (see **RUNTIME MANAGEMENT COMMANDS** below). With this option, **ovsdb−server** instead starts a shell subprocess running *command*. When the subprocess terminates, **ovsdb−server** also exits gracefully. If the subprocess exits normally with exit code 0, then **ovsdb−server** exits with exit code 0 also; otherwise, it exits with exit code 1.

This option can be useful where a database server is needed only to run a single command, e.g.: **ovsdb−server −−remote=punix:socket −−run='ovsdb−client dump unix:socket Open_vSwitch'**

This option is not supported on Windows platform.

Daemon Options
The following options are valid on POSIX based platforms.

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

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

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

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

−−**detach**
Runs **ovsdb−server** 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. **ovsdb−server** detaches only after it starts listening on all configured remotes.

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

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

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

Specifying −−**no−chdir** suppresses this behavior, preventing **ovsdb−server** 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-know, at build-time whitelisted directories. It is better to stick with this default behavior and not to use this flag unless some other Access Control is used to confine daemon. Note that in contrast to other access control implementations that are typically enforced from kernel-space (e.g. DAC or MAC), self-confinement is imposed from the user-space daemon itself and hence should not be considered as a full confinement strategy, but instead should be viewed as an additional layer of security.

−−user Causes **ovsdb−server** 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 two additional capabilities, namely CAP_NET_ADMIN and CAP_NET_RAW. The capability change will apply even if 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.

Service Options
The following options are valid only on Windows platform.

−−service
Causes **ovsdb−server** to run as a service in the background. The service should already have been created through external tools like **SC.exe**.

−−service−monitor
Causes the **ovsdb−server** service to be automatically restarted by the Windows services manager if the service dies or exits for unexpected reasons.

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

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

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

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

Case is not significant within *spec*.

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

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

−v

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

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

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

−−log−file[=*file*]
Enables logging to a file. If *file* is specified, then it is used as the exact name for the log file. The default log file name used if *file* is omitted is **/usr/local/var/log/openvswitch/ovsdb−server.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. This is the default behavior. 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.

Syncing Options
The following options allow **ovsdb−server** to synchronize its databases with another running **ovsdb−server**.

−−sync−from=*server*
Sets up **ovsdb−server** to synchronize its databases with the databases in *server*, which must be an active connection method in one of the forms documented in **ovsdb−client**(1). Every transaction committed by *server* will be replicated to **ovsdb−server**. This option makes **ovsdb−server** start as a backup server; add **−−active** to make it start as an active server.

−−sync−exclude−tables=*db:table[,db:table]...*
Causes the specified tables to be excluded from replication.

−−active
By default, **−−sync−from** makes **ovsdb−server** start up as a backup for *server*. With **−−active**, however, **ovsdb−server** starts as an active server. Use this option to allow the syncing options to be specified using command line options, yet start the server, as the default, active server. To

switch the running server to backup mode, use **ovs-appctl(1)** to execute the **ovsdb−server/connect−active−ovsdb−server** command.

Public Key Infrastructure Options

The options described below for configuring the SSL public key infrastructure accept a special syntax for obtaining their configuration from the database. If any of these options is given **db:***db*,*table*,*column* as its argument, then the actual file name is read from the specified *column* in *table* within the *db* database. The *column* must have type string or set of strings. The first nonempty string in the table is taken as the file name. (This means that ordinarily there should be at most one row in *table*.)

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

Specifies a PEM file containing the private key used as **ovsdb−server**'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 **ovsdb−server** 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 **ovsdb−server** 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 **ovsdb−server**'s own certificate, that is, the certificate specified on **−c** or **−−certificate**. If **ovsdb−server**'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 **ovsdb−server**'s identity. However, this offers a way for a new installation to bootstrap the CA certificate on its first SSL connection.

SSL Connection Options

−−ssl−protocols=*protocols*

Specifies, in a comma- or space-delimited list, the SSL protocols **ovsdb−server** will enable for SSL connections. Supported *protocols* include **TLSv1**, **TLSv1.1**, and **TLSv1.2**. Regardless of

order, the highest protocol supported by both sides will be chosen when making the connection. The default when this option is omitted is **TLSv1,TLSv1.1,TLSv1.2**.

−−ssl−ciphers=*ciphers*
Specifies, in OpenSSL cipher string format, the ciphers **ovsdb−server** will support for SSL connections. The default when this option is omitted is **HIGH:!aNULL:!MD5**.

Other Options

−−unixctl=*socket*
Sets the name of the control socket on which **ovsdb−server** 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/ovsdb−server.***pid***.ctl**, where *pid* is **ovsdb−server**'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 **ovsdb−server.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.

−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 **ovsdb−server** process. The currently supported commands are described below.

OVSDB−SERVER COMMANDS

These commands are specific to **ovsdb−server**.

exit Causes **ovsdb−server** to gracefully terminate.

ovsdb−server/compact [*db*]...
Compacts each database *db* in-place. If no *db* is specified, compacts every database in-place. A database is also compacted automatically when a transaction is logged if it is over 4 times as large as its previous compacted size (and at least 10 MB), but not before 100 commits have been added or 10 minutes have elapsed since the last compaction.

ovsdb−server/reconnect
Makes **ovsdb−server** drop all of the JSON−RPC connections to database clients and reconnect.

This command might be useful for debugging issues with database clients.

ovsdb−server/add−remote *remote*
Adds a remote, as if **−−remote=***remote* had been specified on the **ovsdb−server** command line. (If *remote* is already a remote, this command succeeds without changing the configuration.)

ovsdb−server/remove−remote *remote*
Removes the specified *remote* from the configuration, failing with an error if *remote* is not configured as a remote. This command only works with remotes that were named on **−−remote** or **ovsdb−server/add−remote**, that is, it will not remove remotes added indirectly because they were read from the database by configuring a **db:***db***,***table***,***column* remote. (You can remove a database source with **ovsdb−server/remove−remote db:***db***,***table***,***column*, but not individual remotes found indirectly through the database.)

ovsdb−server/list−remotes
Outputs a list of the currently configured remotes named on **−−remote** or **ovsdb−server/add−remote**, that is, it does not list remotes added indirectly because they were read from the database by configuring a **db:***db***,***table***,***column* remote.

ovsdb−server/add−db *database*
Adds the *database* to the running **ovsdb−server**. The database file must already have been created and initialized using, for example, **ovsdb−tool create**.

ovsdb−server/remove−db *database*
Removes *database* from the running **ovsdb−server**. *database* must be a database name as listed by **ovsdb-server/list−dbs**.

If a remote has been configured that points to the specified *database* (e.g. **−−remote=db:***database***,...** on the command line), then it will be disabled until another database with the same name is added again (with **ovsdb−server/add−db**).

Any public key infrastructure options specified through this database (e.g. **−−private−key=db:***database*,... on the command line) will be disabled until another database with the same name is added again (with **ovsdb−server/add−db**).

ovsdb−server/list−dbs
Outputs a list of the currently configured databases added either through the command line or through the **ovsdb−server/add−db** command.

ovsdb−server/set−active−ovsdb−server *server*
Sets the active *server* from which **ovsdb−server** connects through **ovsdb−server/connect−active−ovsdb−server**.

ovsdb−server/get−active−ovsdb−server
Gets the active server from which **ovsdb−server** is currently synchronizing its databases.

ovsdb−server/connect−active−ovsdb−server
Causes **ovsdb−server** to synchronize its databases with the server specified by **ovsdb−server/set−active−ovsdb−server**.

ovsdb−server/disconnect−active−ovsdb−server
Causes **ovsdb−server** to stop synchronizing its databases with a active server.

ovsdb−server/set−sync−exclude−tables *db***:***table*[**,***db***:***table*]...
Sets the *table* whitin *db* that will be excluded from synchronization.

ovsdb−server/get−sync−exclude−tables
Gets the tables that are currently excluded from synchronization.

ovsdb−server/sync−status
Prints a summary of replication run time information. The **state** information is always provided, indicating whether the server is running in the *active* or the *backup* mode. When running in backup mode, replication connection status, which can be either *connecting*, *replicating* or *error*, are shown. When the connection is in *replicating* state, further output shows the list of databases currently replicating, and the tables that are excluded.

VLOG COMMANDS

These commands manage **ovsdb−server**'s logging settings.

vlog/set [*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.

  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 **ovsdb−server** 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:*destination***:***pattern*
Sets the log pattern for *destination* 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/list-pattern
Lists logging patterns used for each destination.

vlog/close
Causes **ovsdb−server** to close its log file, if it is open. (Use **vlog/reopen** to reopen it later.)

vlog/reopen
Causes **ovsdb−server** to close its log file, if it is open, and then reopen it. (This is useful after rotating log files, to cause a new log file to be used.)

This has no effect unless **ovsdb−server** was invoked with the **−−log−file** option.

vlog/disable−rate−limit [*module*]...
vlog/enable−rate−limit [*module*]...
By default, **ovsdb−server** 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 **ovsdb−server**'s memory usage. **ovsdb−server** also logs this information soon after startup and periodically as its memory consumption grows.

COVERAGE COMMANDS

These commands manage **ovsdb−server**'s ''coverage counters,'' which count the number of times particular events occur during a daemon's runtime. In addition to these commands, **ovsdb−server** 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 averaged per-second rates for the last few seconds, the last minute and the last hour, and the total counts of all of the coverage counters.

SPECIFICATIONS

ovsdb−server implements the Open vSwitch Database (OVSDB) protocol specified in RFC 7047, with the following clarifications:

3.1. JSON Usage

RFC 4627 says that names within a JSON object should be unique. The Open vSwitch JSON parser discards all but the last value for a name that is specified more than once.

The definition of <error> allows for implementation extensions. Currently **ovsdb−server** uses the following additional "error" strings which might change in later releases):

syntax error or **unknown column**
The request could not be parsed as an OVSDB request. An additional "syntax" member, whose value is a string that contains JSON, may narrow down the particular syntax that could not be parsed.

internal error
The request triggered a bug in **ovsdb−server**.

ovsdb error
A map or set contains a duplicate key.

3.2. Schema Format

RFC 7047 requires the "version" field in <database-schema>. Current versions of **ovsdb−server** allow it to be omitted (future versions are likely to require it).

RFC 7047 allows columns that contain weak references to be immutable. This raises the issue of the behavior of the weak reference when the rows that it references are deleted. Since version 2.6, **ovsdb−server** forces columns that contain weak references to be mutable.

4. Wire Protocol

The original OVSDB specifications included the following reason, omitted from RFC 7047, to operate JSON-RPC directly over a stream instead of over HTTP:

- JSON-RPC is a peer-to-peer protocol, but HTTP is a client-server protocol, which is a poor match. Thus, JSON-RPC over HTTP requires the client to periodically poll the server to receive server requests.
- HTTP is more complicated than stream connections and doesn’t provide any corresponding advantage.
- The JSON-RPC specification for HTTP transport is incomplete.

4.1.5. Monitor

For backward compatibility, **ovsdb−server** currently permits a single <monitor-request> to be used instead of an array; it is treated as a single-element array. Future versions of **ovsdb−server** might remove this compatibility feature.

Because the <json-value> parameter is used to match subsequent update notifications (see below) to the request, it must be unique among all active monitors. **ovsdb−server** rejects attempt to create two monitors with the same identifier.

4.1.12. Monitor_cond

A new monitor method added in Open vSwitch version 2.6. The monitor_cond request enables a client to replicate subsets of tables within an OVSDB database by requesting notifications of changes to rows matching one of the conditions specified in "where" by receiving the specified contents of these rows when table updates occur. Monitor_cond also allows a more efficient update notifications by receiving table-updates2 notifications (described below).

The monitor method described in Section 4.1.5 also applies to monitor_cond, with the following exceptions:

- RPC request method becomes "monitor_cond".
- Reply result follows <table-updates2>, described in Section 4.1.14.
- Subsequent changes are sent to the client using the "update2" monitor notification, described in Section 4.1.14
- Update notifications are being sent only for rows matching [<condition>*].

The request object has the following members:

"method": "monitor_cond"
"params": [<db-name>, <json-value>, <monitor-cond-requests>]
"id": <nonnull-json-value>

The <json-value> parameter is used to match subsequent update notifications (see below) to this request. The <monitor-cond-requests> object maps the name of the table to an array of <monitor-cond-request>.

Each <monitor-cond-request> is an object with the following members:

"columns": [<column>*] optional
"where": [<condition>*] optional
"select": <monitor-select> optional

The "columns", if present, define the columns within the table to be monitored that match conditions. If not present all columns are being monitored.

The "where" if present is a JSON array of <condition> and boolean values. If not present or condition is an empty array, implicit True will be considered and updates on all rows will be sent.

<monitor-select> is an object with the following members:

"initial": <boolean> optional
"insert": <boolean> optional
"delete": <boolean> optional
"modify": <boolean> optional

The contents of this object specify how the columns or table are to be monitored as explained in more detail below.

The response object has the following members:

"result": <table-updates2>
"error": null
"id": same "id" as request

The <table-updates2> object is described in detail in Section 4.1.14. It contains the contents of the tables for which "initial" rows are selected. If no tables initial contents are requested, then "result" is an empty object.

Subsequently, when changes to a specified table that match one of the conditions in monitor-cond-request are committed, the changes are automatically sent to the client using the "update2" monitor notification (see Section 4.1.14). This monitoring persists until the JSON-RPC session terminates or until the client sends a "monitor_cancel" JSON-RPC request.

Each <monitor-cond-request> specifies one or more conditions and the manner in which the rows that match the conditions are to be monitored. The circumstances in which an "update" notification is sent for a row within the table are determined by <monitor-select>:

- If "initial" is omitted or true, every row in the original table that matches one of the conditions is sent as part of the response to the "monitor_cond" request.

- If "insert" is omitted or true, "update" notifications are sent for rows newly inserted into the table that match conditions or for rows modified in the table so that their old version does not match the condition and new version does.
- If "delete" is omitted or true, "update" notifications are sent for rows deleted from the table that match conditions or for rows modified in the table so that their old version does match the conditions and new version does not.
- If "modify" is omitted or true, "update" notifications are sent whenever a row in the table that matches conditions in both old and new version is modified.

Both monitor and monitor_cond sessions can exist concurrently. However, monitor and monitor_cond shares the same <json-value> parameter space; it must be unique among all monitor and monitor_cond sessions.

4.1.13. Monitor_cond_change

The "monitor_cond_change" request enables a client to change an existing "monitor_cond" replication of the database by specifying a new condition and columns for each replicated table. Currently changing the columns set is not supported.

The request object has the following members:

"method": "monitor_cond_change"
"params": [<json-value>, <json-value>, <monitor-cond-update-requests>]
"id": <nonnull-json-value>

The <json-value> parameter should have a value of an existing conditional monitoring session from this client. The second <json-value> in params array is the requested value for this session. This value is valid only after "monitor_cond_change" is committed. A user can use these values to distinguish between update messages before conditions update and after. The <monitor-cond-update-requests> object maps the name of the table to an array of <monitor-cond-update-request>.

Each <monitor-cond-update-request> is an object with the following members:

"columns": [<column>*] optional
"where": [<condition>*] optional

The "columns" specify a new array of columns to be monitored (Currently unsupported).

The "where" specify a new array of conditions to be applied to this monitoring session.

The response object has the following members:

"result": null
"error": null
"id": same "id" as request

Subsequent <table-updates2> notifications are described in detail in Section 4.1.14 in the RFC. If insert contents are requested by original monitor_cond request, <table-updates2> will contain rows that match the new condition and do not match the old condition. If deleted contents are requested by origin monitor request, <table-updates2> will contain any matched rows by old condition and not matched by the new condition.

Changes according to the new conditions are automatically sent to the client using the "update2" monitor notification. An update, if any, as a result of a condition change, will be sent to the client before the reply to the "monitor_cond_change" request.

4.1.14. Update2 notification

The "update2" notification is sent by the server to the client to report changes in tables that are being monitored following a "monitor_cond" request as described above. The notification has the following members:
"method": "update2"
"params": [<json-value>, <table-updates2>]
"id": null

The <json-value> in "params" is the same as the value passed as the <json-value> in "params" for the corresponding "monitor" request. <table-updates2> is an object that maps from a table name to a <table-update2>. A <table-update2> is an object that maps from row's UUID to a <row-update2> object. A <row-update2> is an object with one of the following members:

"initial": <row>
present for "initial" updates

"insert": <row>
present for "insert" updates

"delete": <row>
present for "delete" updates

"modify": <row>
present for "modify" updates

The format of <row> is described in Section 5.1.

<row> is always a null object for a "delete" update. In "initial" and "insert" updates, <row> omits columns whose values equal the default value of the column type.

For a "modify" update, <row> contains only the columns that are modified. <row> stores the difference between the old and new value for those columns, as described below.

For columns with single value, the difference is the value of the new column.

The difference between two sets are all elements that only belong to one of the sets.

The difference between two maps are all key-value pairs whose keys appears in only one of the maps, plus the key-value pairs whose keys appear in both maps but with different values. For the latter elements, <row> includes the value from the new column.

Initial views of rows are not presented in update2 notifications, but in the response object to the monitor_cond request. The formatting of the <table-updates2> object, however, is the same in either case.

4.1.15. Get Server ID

A new RPC method added in Open vSwitch version 2.7. The request contains the following members:

"method": "get_server_id"
"params": null
"id": <nonnull-json-value>

The response object contains the following members:

"result": "<server_id>"
"error": null
"id": same "id" as request

<server_id> is JSON string that contains a UUID that uniquely identifies the running OVSDB server process. A fresh UUID is generated when the process restarts.

5.1. Notation

For <condition>, RFC 7047 only allows the use of **!=**, **==**, **includes**, and **excludes** operators with set types. Open vSwitch 2.4 and later extend <condition> to allow the use of **<**, **<=**, **>=**, and **>** operators with columns with type ''set of 0 or 1 integer'' and ''set of 0 or 1 real''. These conditions evaluate to false when the column is empty, and otherwise as described in RFC 7047 for integer and real types.

<condition> is specified in Section 5.1 in the RFC with the following change: A condition can be either a 3-element JSON array as described in the RFC or a boolean value. In case of an empty array an implicit true boolean value will be considered.

BUGS

In Open vSwitch before version 2.4, when **ovsdb−server** sent JSON-RPC error responses to some requests, it incorrectly formulated them with the **result** and **error** swapped, so that the response appeared to indicate success (with a nonsensical result) rather than an error. The requests that suffered from this problem were:

transact
get_schema
: Only if the request names a nonexistent database.

monitor
lock
unlock In all error cases.

Of these cases, the only error that a well-written application is likely to encounter in practice is **monitor** of tables or columns that do not exist, in an situation where the application has been upgraded but the old database schema is still temporarily in use. To handle this situation gracefully, we recommend that clients should treat a **monitor** response with a **result** that contains an **error** key-value pair as an error (assuming that the database being monitored does not contain a table named **error**).

SEE ALSO

ovsdb−tool(1).