NAME
ovn−sbctl – utility for querying and configuring OVN_Southbound database

SYNOPSIS
ovn−sbctl [options] command [args] [−− [options] command [args]]...

DESCRIPTION
The command should only be used for advanced debugging and troubleshooting of the OVN_Southbound database; and should never be used in normal operation.

The ovn−sbctl program configures the OVN_Southbound database by providing a high−level interface to its configuration database. See ovn−sb(5) for comprehensive documentation of the database schema.

ovn−sbctl connects to an ovsvdb−server process that maintains an OVN_Southbound configuration database. Using this connection, it queries and possibly applies changes to the database, depending on the supplied commands.

ovn−sbctl can perform any number of commands in a single run, implemented as a single atomic transaction against the database.

The ovn−sbctl command line begins with global options (see OPTIONS below for details). The global options are followed by one or more commands. Each command should begin with −− by itself as a command-line argument, to separate it from the following commands. (The −− before the first command is optional.) The command itself starts with command-specific options, if any, followed by the command name and any arguments.

OPTIONS
The following options affect the behavior of ovn−sbctl as a whole. Some individual commands also accept their own options, which are given just before the command name. If the first command on the command line has options, then those options must be separated from the global options by −−.

−−db=server
The OVSDB database remote to contact. If the OVNSB_DB environment variable is set, its value is used as the default. Otherwise, the default is unix:/var/run/openvswitch/db.sock, but this default is unlikely to be useful outside of single-machine OVN test environments.

server must take one of the following forms:

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 localhost TCP port whose value is written in 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 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 \textit{ip}, either IPv4 or IPv6 address. If \textit{ip} is an IPv6 address, then wrap \textit{ip} with square brackets, e.g.: \texttt{ptcp:6640:[::1]}.

\texttt{punix:file}

On POSIX, listen on the Unix domain server socket named \textit{file} for a connection.

On Windows, listen on a kernel chosen TCP port on the localhost. The kernel chosen TCP port value is written in \textit{file}.

\texttt{--no-syslog}

By default, \texttt{ovn-sbctl} logs its arguments and the details of any changes that it makes to the system log. This option disables this logging.

This option is equivalent to \texttt{--verbose=sbctl:syslog:warn}.

\texttt{--oneline}

Modifies the output format so that the output for each command is printed on a single line. Newline characters that would otherwise separate lines are printed as \texttt{\textbackslash n}, and any instances of \texttt{\textbackslash} that would otherwise appear in the output are doubled. Prints a blank line for each command that has no output. This option does not affect the formatting of output from the \texttt{list} or \texttt{find} commands; see Table Formatting Options below.

\texttt{--dry-run}

Prevents \texttt{ovn-sbctl} from actually modifying the database.

\texttt{--timeout=secs}

By default, or with a \texttt{secs} of 0, \texttt{ovn-sbctl} waits forever for a response from the database. This option limits runtime to approximately \texttt{secs} seconds. If the timeout expires, \texttt{ovn-sbctl} will exit with a \texttt{SIGALRM} signal. (A timeout would normally happen only if the database cannot be contacted, or if the system is overloaded.)

Table Formatting Options

These options control the format of output from the \texttt{list} and \texttt{find} commands.

\texttt{--format=format}

Sets the type of table formatting. The following types of \texttt{format} are available:

\begin{itemize}
  \item \texttt{table} (default)
    \begin{itemize}
      \item 2-D text tables with aligned columns.
    \end{itemize}
  \item \texttt{list}
    \begin{itemize}
      \item A list with one column per line and rows separated by a blank line.
    \end{itemize}
  \item \texttt{html}
    \begin{itemize}
      \item HTML tables.
    \end{itemize}
  \item \texttt{csv}
    \begin{itemize}
      \item Comma-separated values as defined in RFC 4180.
    \end{itemize}
  \item \texttt{json}
    \begin{itemize}
      \item JSON format as defined in RFC 4627. The output is a sequence of JSON objects, each of which corresponds to one table. Each JSON object has the following members with the noted values:
        \begin{itemize}
          \item \texttt{caption}
            \begin{itemize}
              \item The table’s caption. This member is omitted if the table has no caption.
            \end{itemize}
          \item \texttt{headings}
            \begin{itemize}
              \item An array with one element per table column. Each array element is a string giving the corresponding column’s heading.
            \end{itemize}
          \item \texttt{data}
            \begin{itemize}
              \item An array with one element per table row. Each element is also an array with one element per table column. The elements of this second-level array are the cells that constitute the table. Cells that represent OVSDB data or data types are expressed in the format described in the OVSDB specification; other cells are simply expressed as text strings.
            \end{itemize}
        \end{itemize}
    \end{itemize}
\end{itemize}
ovn−sbctl(8) Open vSwitch Manual  ovn−sbctl(8)

−d format
−−data=format
Sets the formatting for cells within output tables. The following types of format are available:

string (default)
The simple format described in the Database Values section of ovs−vsctl(8).
bare The simple format with punctuation stripped off: [] and {} are omitted around sets, maps, and empty columns, items within sets and maps are space-separated, and strings are never quoted. This format may be easier for scripts to parse.
json JSON.
The json output format always outputs cells in JSON format, ignoring this option.

−−no−heading
This option suppresses the heading row that otherwise appears in the first row of table output.

−−pretty
By default, JSON in output is printed as compactly as possible. This option causes JSON in output to be printed in a more readable fashion. Members of objects and elements of arrays are printed one per line, with indentation.
This option does not affect JSON in tables, which is always printed compactly.

−−bare Equivalent to −−format=list −−data=bare −−no−headings.

COMMANDS
The commands implemented by ovn−sbctl are described in the sections below.

OVN_Southbound Commands
These commands work with an OVN_Southbound database as a whole.

show Prints a brief overview of the database contents.

Chassis Commands
These commands manipulate OVN_Southbound chassis.

[−−may−exist] chassis−add chassis encap-type encap-ip
Creates a new chassis named chassis. The chassis will have one encap entry with encap-type as tunnel type and encap-ip as destination ip.
Without −−may−exist, attempting to create a chassis that exists is an error. With −−may−exist, this command does nothing if chassis already exists as a real bridge.

[−−if−exists] chassis−del chassis
Deletes chassis and its encaps and gateway_ports.
Without −−if−exists, attempting to delete a chassis that does not exist is an error. With −−if−exists, attempting to delete a chassis that does not exist has no effect.

Port binding Commands
These commands manipulate OVN_Southbound port bindings.

[−−may−exist] lport−bind logical−port chassis
Binds the logical port named logical−port to chassis.
Without −−may−exist, attempting to bind a logical port that has already been bound is an error. With −−may−exist, this command does nothing if logical−port has already been bound to a chassis.

[−−if−exists] lport−unbind logical−port
Resets the binding of logical−port to NULL.
Without −−if−exists, attempting to unbind a logical port that is not bound is an error. With −−if−exists, attempting to unbind logical port that is not bound has no effect.
Logical Flow Commands

lflow-list [logical−datapath]
List logical flows. If logical−datapath is specified, only list flows for that logical datapath.

dump−flows [logical−datapath]
Alias for lflow-list.

Database Values

Each column in the database accepts a fixed type of data. The currently defined basic types, and their representations, are:

- integer: A decimal integer in the range $-2^{63}$ to $2^{63}-1$, inclusive.
- real: A floating-point number.
- Boolean: True or false, written true or false, respectively.
- string: An arbitrary Unicode string, except that null bytes are not allowed. Quotes are optional for most strings that begin with an English letter or underscore and consist only of letters, underscores, hyphens, and periods. However, true and false and strings that match the syntax of UUIDs (see below) must be enclosed in double quotes to distinguish them from other basic types. When double-quotes are used, the syntax is that of strings in JSON, e.g. backslashes may be used to escape special characters. The empty string must be represented as a pair of double quotes (""").
- UUID: Either a universally unique identifier in the style of RFC 4122, e.g. f81d4fae−7dec−11d0−a765−00a0c91e6bf6, or an @name defined by a get or create command within the same ovn−sbctl invocation.

Multiple values in a single column may be separated by spaces or a single comma. When multiple values are present, duplicates are not allowed, and order is not important. Conversely, some database columns can have an empty set of values, represented as [], and square brackets may optionally enclose other non-empty sets or single values as well.

A few database columns are “maps” of key-value pairs, where the key and the value are each some fixed database type. These are specified in the form key=value, where key and value follow the syntax for the column’s key type and value type, respectively. When multiple pairs are present (separated by spaces or a comma), duplicate keys are not allowed, and again the order is not important. Duplicate values are allowed. An empty map is represented as {}. Curly braces may optionally enclose non-empty maps as well (but use quotes to prevent the shell from expanding other-config={0=x,1=y} into other-config=0=x other-config=1=y, which may not have the desired effect).

Database Command Syntax

[−−if-exists] [−−columns=column[,column]...] list table [record]...
Lists the data in each specified record. If no records are specified, lists all the records in table.

If −−columns is specified, only the requested columns are listed, in the specified order. Otherwise, all columns are listed, in alphabetical order by column name.

Without −−if-exists, it is an error if any specified record does not exist. With −−if-exists, the command ignores any record that does not exist, without producing any output.

[−−columns=column[,column]...] find table [column:[key]=value]...
Lists the data in each record in table whose column equals value or, if key is specified, whose column contains a key with the specified value. The following operators may be used where = is written in the syntax summary:

- !=: Selects records in which column[key] equals, does not equal, less than, is greater than, is less than or equal to, or is greater than or equal to value, respectively.

- Consider column[key] and value as sets of elements. Identical sets are considered equal. Otherwise, if the sets have different numbers of elements, then the set with more elements
is considered to be larger. Otherwise, consider a element from each set pairwise, in
increasing order within each set. The first pair that differs determines the result. (For a
column that contains key-value pairs, first all the keys are compared, and values are con-
sidered only if the two sets contain identical keys.)

\{\} \{!\} =
Test for set equality or inequality, respectively.

\{<=\} Selects records in which column[;key] is a subset of value. For example, flood-
vlans[<=]1,2 selects records in which the flood-vlans column is the empty set or con-
tains 1 or 2 or both.

\{<\} Selects records in which column[;key] is a proper subset of value. For example, flood-
vlans[<]1,2 selects records in which the flood-vlans column is the empty set or contains
1 or 2 but not both.

\{>=\} \{>\} Same as \{<=\} and \{<\}, respectively, except that the relationship is reversed. For example, 
flood-vlans[>=]1,2 selects records in which the flood-vlans column contains both 1 and
2.

For arithmetic operators (\{\}\{!\} < > <= >= =), when key is specified but a particular record’s column
does not contain key, the record is always omitted from the results. Thus, the condition other-con-
fig:mtu!=1500 matches records that have a mtu key whose value is not 1500, but not those that
lack an mtu key.

For the set operators, when key is specified but a particular record’s column does not contain key,
the comparison is done against an empty set. Thus, the condition other-config:mtu!(\{\}1500
matches records that have a mtu key whose value is not 1500 and those that lack an mtu key.

Don’t forget to escape < or > from interpretation by the shell.

If \{--columns\} is specified, only the requested columns are listed, in the specified order. Otherwise
all columns are listed, in alphabetical order by column name.

The UUIDs shown for rows created in the same ovn-sbctl invocation will be wrong.

\{--if-exists\} [\{--id=@name\}] get table record [column[;key]]...
Prints the value of each specified column in the given record in table. For map columns, a key
may optionally be specified, in which case the value associated with key in the column is printed,
instead of the entire map.

Without \{--if-exists\}, it is an error if record does not exist or key is specified, if key does not exist
in record. With \{--if-exists\}, a missing record yields no output and a missing key prints a blank
line.

If @name is specified, then the UUID for record may be referred to by that name later in the same
ovn-sbctl invocation in contexts where a UUID is expected.

Both \{--id\} and the \{column\} arguments are optional, but usually at least one or the other should be
specified. If both are omitted, then \{get\} has no effect except to verify that record exists in table.

\{--id\} and \{--if-exists\} cannot be used together.

\{--if-exists\} set table record column[;key]=value...
Sets the value of each specified column in the given record in table to value. For map columns, a key
may optionally be specified, in which case the value associated with key in that column is
changed (or added, if none exists), instead of the entire map.

Without \{--if-exists\}, it is an error if record does not exist. With \{--if-exists\}, this command does
nothing if record does not exist.

\{--if-exists\} add table record column [key]=value...
Adds the specified value or key-value pair to column in record in table. If column is a map, then
key is required, otherwise it is prohibited. If key already exists in a map column, then the current
value is not replaced (use the set command to replace an existing value).

Without --if-exists, it is an error if record does not exist. With --if-exists, this command does nothing if record does not exist.

[--if-exists] remove table record column value...
[--if-exists] remove table record column key...
[--if-exists] remove table record column key=value...

Removes the specified values or key-value pairs from column in record in table. The first form applies to columns that are not maps: each specified value is removed from the column. The second and third forms apply to map columns: if only a key is specified, then any key-value pair with the given key is removed, regardless of its value; if a value is given then a pair is removed only if both key and value match.

It is not an error if the column does not contain the specified key or value or pair.

Without --if-exists, it is an error if record does not exist. With --if-exists, this command does nothing if record does not exist.

[--if-exists] clear table record column...

Sets each column in record in table to the empty set or empty map, as appropriate. This command applies only to columns that are allowed to be empty.

Without --if-exists, it is an error if record does not exist. With --if-exists, this command does nothing if record does not exist.

[--id=@name] create table column[;key]=value...

Creates a new record in table and sets the initial values of each column. Columns not explicitly set will receive their default values. Outputs the UUID of the new row.

If @name is specified, then the UUID for the new row may be referred to by that name elsewhere in the same ovn--sbctl invocation in contexts where a UUID is expected. Such references may precede or follow the create command.

Caution (ovs-vsctl as example)

Records in the Open vSwitch database are significant only when they can be reached directly or indirectly from the Open_vSwitch table. Except for records in the QoS or Queue tables, records that are not reachable from the Open_vSwitch table are automatically deleted from the database. This deletion happens immediately, without waiting for additional ovs-vsctl commands or other database activity. Thus, a create command must generally be accompanied by additional commands within the same ovs-vsctl invocation to add a chain of references to the newly created record from the top-level Open_vSwitch record. The EXAMPLES section gives some examples that show how to do this.

[--if-exists] destroy table record...

Deletes each specified record from table. Unless --if-exists is specified, each records must exist.

--all destroy table

Deletes all records from the table.

Caution (ovs-vsctl as example)

The destroy command is only useful for records in the QoS or Queue tables. Records in other tables are automatically deleted from the database when they become unreachable from the Open_vSwitch table. This means that deleting the last reference to a record is sufficient for deleting the record itself. For records in these tables, destroy is silently ignored. See the EXAMPLES section below for more information.

wait-until table record [column[;key]=value]...

Waits until table contains a record named record whose column equals value or, if key is specified, whose column contains a key with the specified value. Any of the operators !=, <, >, <=, or >= may be substituted for = to test for inequality, less than, greater than, less than or equal to, or
greater than or equal to, respectively. (Don’t forget to escape < or > from interpretation by the shell.)

If no column[key]=value arguments are given, this command waits only until record exists. If more than one such argument is given, the command waits until all of them are satisfied.

Caution (ovs-vsctl as example)

Usually `wait-until` should be placed at the beginning of a set of `ovs-vsctl` commands. For example, `wait-until bridge br0 -- get bridge br0 datapath_id` waits until a bridge named `br0` is created, then prints its `datapath_id` column, whereas `get bridge br0 datapath_id -- wait-until bridge br0` will abort if no bridge named `br0` exists when `ovs-vsctl` initially connects to the database.

Consider specifying `--timeout=0` along with `--wait-until`, to prevent `ovn-sbctl` from terminating after waiting only at most 5 seconds.

```
comment [arg]...
```

This command has no effect on behavior, but any database log record created by the command will include the command and its arguments.

**EXIT STATUS**

```
0  Successful program execution.
1  Usage, syntax, or configuration file error.
2  The bridge argument to `br-exists` specified the name of a bridge that does not exist.
```

**SEE ALSO**

`ovn-sb(5)`. 