NAME
ovn-nbctl – Open Virtual Network northbound db management utility

SYNOPSYS
ovn−nbctl [*options*] *command* [*arg...*]

DESCRIPTION
This utility can be used to manage the OVN northbound database.

GENERAL COMMANDS
show [*lswitch***]**

Prints a brief overview of the database contents. If *lswitch* is provided, only records related to that logical switch are shown.

LOGICAL SWITCH COMMANDS
lswitch−add [*lswitch*]

Creates a new logical switch named *lswitch*. If *lswitch* is not provided, the switch will not have a name so other commands must refer to this switch by its UUID. Initially the switch will have no ports.

lswitch−del *lswitch*

Deletes *lswitch*.

lswitch−list

Lists all existing switches on standard output, one per line.

ACL COMMANDS
[**−−log**] **acl−add** *lswitch direction priority match action*

Adds the specified ACL to *lswitch*. *direction* must be either **from−lport** or **to−lport**. *priority* must be between **1** and **65534**, inclusive. If **−−log** is specified, packet logging is enabled for the ACL. A full description of the fields are in **ovn−nb**(5).

acl−del *lswitch* [*direction* [*priority match*]]

Deletes ACLs from *lswitch*. If only *lswitch* is supplied, all the ACLs from the logical switch are deleted. If *direction* is also specified, then all the flows in that direction will be deleted from the logical switch. If all the fields are given, then a single flow that matches all the fields will be deleted.

acl−list *lswitch*

Lists the ACLs on *lswitch*.

LOGICAL PORT COMMANDS
lport−add *lswitch lport*

Creates on *lswitch* a new logical port named *lport*.

lport−add *lswitch lport parent tag*

Creates on *lswitch* a logical port named *lport* that is a child of *parent* that is identifed with VLAN ID *tag*. This is useful in cases such as virtualized container environments where Open vSwitch does not have a direct connection to the container's port and it must be shared with the virtual machine's port.

lport−del *lport*

Deletes *lport*.

lport−list *lswitch*

Lists all the logical ports within *lswitch* on standard output, one per line.

lport−get−parent *lport*

If set, get the parent port of *lport*. If not set, print nothing.

lport−get−tag *lport*

If set, get the tag for *lport* traffic. If not set, print nothing.

lport−set−addresses *lport* [*address*]...

      Sets the addresses associated with *lport* to *address*. Each *address* should be either an Ethernet address or an Ethernet address followed by an IP address (separated by a space and quoted to form a single command-line argument). The special form **unknown** is also valid. Multiple Ethernet addresses or Ethernet+IP pairs may be set. If no *address* argument is given, *lport* will have no addresses associated with it.

lport−get−addresses *lport*

      Lists all the addresses associated with *lport* on standard output, one per line.

lport−set−port−security *lport* [*addrs*]...

      Sets the port security addresses associated with *lport* to *addrs*. Multiple sets of addresses may be set by using multiple *addrs* arguments. If no *addrs* argument is given, *lport* will not have port security enabled.

      Port security limits the addresses from which a logical port may send packets and to which it may receive packets. See the **ovn−nb**(5) documentation for the **port_security** column in the **Logical_Port** table for details.

lport−get−port−security *lport*

      Lists all the port security addresses associated with *lport* on standard output, one per line.

lport−get−up *lport*

      Prints the state of *lport*, either **up** or **down**.

lport−set−enabled *lport state*

      Set the administrative state of *lport*, either **enabled** or **disabled**. When a port is disabled, no traffic is allowed into or out of the port.

lport−get−enabled *lport*

      Prints the administrative state of *lport*, either **enabled** or **disabled**.

lport−set−type *lport type*

      Set the type for the logical port. No special types have been implemented yet.

lport−get−type *lport*

      Get the type for the logical port.

lport−set−options *lport* [*key=value*]...

      Set type-specific key-value options for the logical port.

lport−get−options *lport*

      Get the type-specific options for the logical port.

DATABASE COMMANDS

These commands query and modify the contents of **ovsdb** tables. They are a slight abstraction of the **ovsdb** interface and as suchthey operate at a lower level than other **ovn−nbctl** commands.

Identifying Tables, Records, and Columns

Each of these commands has a *table* parameter to identify a table within the database. Many of them also take a *record* parameter that identifies a particular record within a table. The *record* parameter may be the UUID for a record, and many tables offer additional ways to identify records. Some commands also take *column* parameters that identify a particular field within the records in a table.

The following tables are currently defined:

Logical_Switch

      An L2 logical switch. Records may be identified by name.

Logical_Port

      A port within an L2 logical switch. Records may be identified by name.

ACL   An ACL rule for a logical switch that points to it through its *acls* column.

Logical_Router

An L3 logical router. Records may be identified by name.

Logical_Router_Port

A port within an L3 logical router. Records may be identified by name.

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−nbctl** 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, is 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 considered 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 contains 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−config: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−nbctl** 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−nbctl** 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**] **remov** *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 ***(PN** invocation in contexts where a UUID is expected. Such references may precede or follow the **create** command.

Caution (ovs-vsctl as exmaple)
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 *record*s must exist.

−−all destroy *table*

> Deletes all records from the *table*.

> Caution (ovs-vsctl as exmaple)
>> 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 exmaple)
>> 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−nbctl** 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.

OPTIONS

−−db *database*

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

−h | **−−help**
−o | **−−options**
−V | **−−version**

LOGGING OPTIONS

−v*spec*, **−−verbose=***spec*
−v, **−−verbose**
−−log−file[=*file*]
−−syslog−target=*host***:***port*

PKI CONFIGURATION (REQUIRED TO USE SSL)

−p, **−−private−key=***file*   file with private key
−c, **−−certificate=***file*   file with certificate for private key
−C, **−−ca−cert=***file*   file with peer CA certificate