

NAME

hardware_vtep – hardware_vtep database schema

This schema specifies relations that a VTEP can use to integrate physical ports into logical switches maintained by a network virtualization controller such as NVP.

Glossary:

VTEP	VXLAN Tunnel End Point, an entity which originates and/or terminates VXLAN tunnels.
HSC	Hardware Switch Controller.
NVC	Network Virtualization Controller, e.g. NVP.
VRF	Virtual Routing and Forwarding instance.

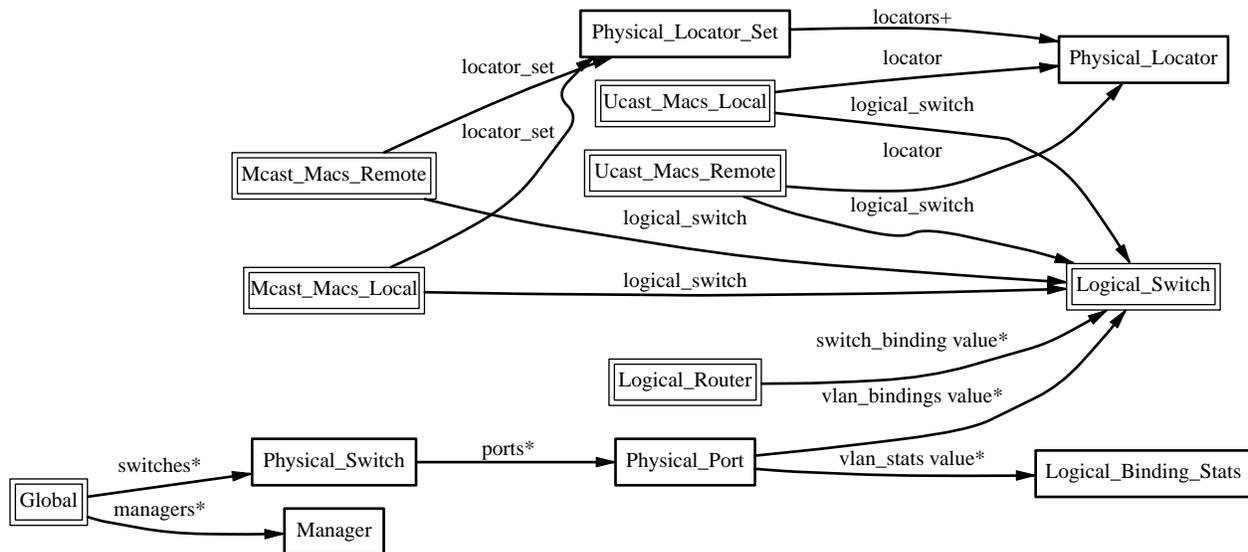
TABLE SUMMARY

The following list summarizes the purpose of each of the tables in the **hardware_vtep** database. Each table is described in more detail on a later page.

Table	Purpose
Global	Top-level configuration.
Manager	OVSDB management connection.
Physical_Switch	A physical switch.
Physical_Port	A port within a physical switch.
Logical_Binding_Stats	Statistics for a VLAN on a physical port bound to a logical network.
Logical_Switch	A layer-2 domain.
Ucast_Macs_Local	Unicast MACs (local)
Ucast_Macs_Remote	Unicast MACs (remote)
Mcast_Macs_Local	Multicast MACs (local)
Mcast_Macs_Remote	Multicast MACs (remote)
Logical_Router	A logical L3 router.
Physical Locator_Set	Physical_Locator_Set configuration.
Physical_Locator	Physical_Locator configuration.

TABLE RELATIONSHIPS

The following diagram shows the relationship among tables in the database. Each node represents a table. Tables that are part of the “root set” are shown with double borders. Each edge leads from the table that contains it and points to the table that its value represents. Edges are labeled with their column names, followed by a constraint on the number of allowed values: ? for zero or one, * for zero or more, + for one or more. Thick lines represent strong references; thin lines represent weak references.



Global TABLE

Top-level configuration for a hardware VTEP. There must be exactly one record in the **Global** table.

Summary:

switches set of **Physical_Switchs**

Database Configuration:

managers set of **Managers**

Details:

switches: set of **Physical_Switchs**

The physical switches managed by the VTEP.

Database Configuration:

These columns primarily configure the Open vSwitch database (**ovsdb-server**), not the hardware VTEP itself.

managers: set of **Managers**

Database clients to which the Open vSwitch database server should connect or to which it should listen, along with options for how these connection should be configured. See the **Manager** table for more information.

Manager TABLE

Configuration for a database connection to an Open vSwitch database (OVSDB) client.

The Open vSwitch database server can initiate and maintain active connections to remote clients. It can also listen for database connections.

Summary:

Core Features:

target string (must be unique within table)

Client Failure Detection and Handling:

max_backoff optional integer, at least 1,000

inactivity_probe optional integer

Status:

is_connected boolean

status : last_error optional string

status : state optional string, one of **ACTIVE**, **VOID**, **CONNECTING**, **IDLE**, or **BACKOFF**

status : sec_since_connect optional string, containing an integer, at least 0

status : sec_since_disconnect optional string, containing an integer, at least 0

status : locks_held optional string

status : locks_waiting optional string

status : locks_lost optional string

status : n_connections optional string, containing an integer, at least 2

Connection Parameters:

other_config : dscp optional string, containing an integer

Details:

Core Features:

target: string (must be unique within table)

Connection method for managers.

The following connection methods are currently supported:

ssl:ip[:port]

The specified SSL *port* (default: 6632) on the host at the given *ip*, which must be expressed as an IP address (not a DNS name).

SSL key and certificate configuration happens outside the database.

tcp:ip[:port]

The specified TCP *port* (default: 6632) on the host at the given *ip*, which must be expressed as an IP address (not a DNS name).

pssl:[port][:ip]

Listens for SSL connections on the specified TCP *port* (default: 6632). If *ip*, which must be expressed as an IP address (not a DNS name), is specified, then connections are restricted to the specified local IP address.

ptcp:[port][:ip]

Listens for connections on the specified TCP *port* (default: 6632). If *ip*, which must be expressed as an IP address (not a DNS name), is specified, then connections are restricted to the specified local IP address.

Client Failure Detection and Handling:

max_backoff: optional integer, at least 1,000

Maximum number of milliseconds to wait between connection attempts. Default is implementation-specific.

inactivity_probe: optional integer

Maximum number of milliseconds of idle time on connection to the client before sending an inactivity probe message. If Open vSwitch does not communicate with the client for the specified number of seconds, it will send a probe. If a response is not received for the same additional amount of time, Open vSwitch assumes the connection has been broken and attempts to reconnect. Default is implementation-specific. A value of 0 disables inactivity probes.

Status:

is_connected: boolean

true if currently connected to this manager, **false** otherwise.

status : last_error: optional string

A human-readable description of the last error on the connection to the manager; i.e. **strerror(errno)**. This key will exist only if an error has occurred.

status : state: optional string, one of **ACTIVE**, **VOID**, **CONNECTING**, **IDLE**, or **BACKOFF**

The state of the connection to the manager:

VOID Connection is disabled.

BACKOFF

Attempting to reconnect at an increasing period.

CONNECTING

Attempting to connect.

ACTIVE

Connected, remote host responsive.

IDLE Connection is idle. Waiting for response to keep-alive.

These values may change in the future. They are provided only for human consumption.

status : sec_since_connect: optional string, containing an integer, at least 0

The amount of time since this manager last successfully connected to the database (in seconds). Value is empty if manager has never successfully connected.

status : sec_since_disconnect: optional string, containing an integer, at least 0

The amount of time since this manager last disconnected from the database (in seconds). Value is empty if manager has never disconnected.

status : locks_held: optional string

Space-separated list of the names of OVSDB locks that the connection holds. Omitted if the connection does not hold any locks.

status : locks_waiting: optional string

Space-separated list of the names of OVSDB locks that the connection is currently waiting to acquire. Omitted if the connection is not waiting for any locks.

status : locks_lost: optional string

Space-separated list of the names of OVSDB locks that the connection has had stolen by another OVSDB client. Omitted if no locks have been stolen from this connection.

status : n_connections: optional string, containing an integer, at least 2

When **target** specifies a connection method that listens for inbound connections (e.g. **ptcp**: or **pssl**:) and more than one connection is actually active, the value is the number of active connections. Otherwise, this key-value pair is omitted.

When multiple connections are active, status columns and key-value pairs (other than this one) report the status of one arbitrarily chosen connection.

Connection Parameters:

Additional configuration for a connection between the manager and the Open vSwitch Database.

other_config : dscp: optional string, containing an integer

The Differentiated Service Code Point (DSCP) is specified using 6 bits in the Type of Service (TOS) field in the IP header. DSCP provides a mechanism to classify the network traffic and provide Quality of Service (QoS) on IP networks. The DSCP value specified here is used when establishing the connection between the manager and the Open vSwitch. If no value is specified, a default value of 48 is chosen. Valid DSCP values must be in the range 0 to 63.

Physical_Switch TABLE

A physical switch that implements a VTEP.

Summary:

ports	set of Physical_Ports
<i>Network Status:</i>	
management_ips	set of strings
tunnel_ips	set of strings
<i>Identification:</i>	
name	string (must be unique within table)
description	string

Details:

ports: set of **Physical_Ports**
The physical ports within the switch.

Network Status:

management_ips: set of strings
IPv4 or IPv6 addresses at which the switch may be contacted for management purposes.

tunnel_ips: set of strings
IPv4 or IPv6 addresses on which the switch may originate or terminate tunnels.

This column is intended to allow a **Manager** to determine the **Physical_Switch** that terminates the tunnel represented by a **Physical_Locator**.

Identification:

name: string (must be unique within table)
Symbolic name for the switch, such as its hostname.

description: string
An extended description for the switch, such as its switch login banner.

Physical_Port TABLE

A port within a **Physical_Switch**.

Summary:

vlan_bindings	map of integer- Logical_Switch pairs, key in range 0 to 4,095
vlan_stats	map of integer- Logical_Binding_Stats pairs, key in range 0 to 4,095

Identification:

name	string
description	string

Details:

vlan_bindings: map of integer-**Logical_Switch** pairs, key in range 0 to 4,095

Identifies how VLANs on the physical port are bound to logical switches. If, for example, the map contains a (VLAN, logical switch) pair, a packet that arrives on the port in the VLAN is considered to belong to the paired logical switch.

vlan_stats: map of integer-**Logical_Binding_Stats** pairs, key in range 0 to 4,095

Statistics for VLANs bound to logical switches on the physical port. An implementation that fully supports such statistics would populate this column with a mapping for every VLAN that is bound in **vlan_bindings**. An implementation that does not support such statistics or only partially supports them would not populate this column or partially populate it, respectively.

Identification:

name: string

Symbolic name for the port. The name ought to be unique within a given **Physical_Switch**, but the database is not capable of enforcing this.

description: string

An extended description for the port.

Logical_Binding_Stats TABLE

Reports statistics for the **Logical_Switch** with which a VLAN on a **Physical_Port** is associated.

Summary:

Statistics:

packets_from_local	integer
bytes_from_local	integer
packets_to_local	integer
bytes_to_local	integer

Details:

Statistics:

These statistics count only packets to which the binding applies.

packets_from_local: integer

Number of packets sent by the **Physical_Switch**.

bytes_from_local: integer

Number of bytes in packets sent by the **Physical_Switch**.

packets_to_local: integer

Number of packets received by the **Physical_Switch**.

bytes_to_local: integer

Number of bytes in packets received by the **Physical_Switch**.

Logical_Switch TABLE

A logical Ethernet switch, whose implementation may span physical and virtual media, possibly crossing L3 domains via tunnels; a logical layer-2 domain; an Ethernet broadcast domain.

Summary:

Per Logical-Switch Tunnel Key:

tunnel_key	optional integer
<i>Identification:</i>	
name	string (must be unique within table)
description	string

Details:

Per Logical-Switch Tunnel Key:

Tunnel protocols tend to have a field that allows the tunnel to be partitioned into sub-tunnels: VXLAN has a VNI, GRE and STT have a key, CAPWAP has a WSI, and so on. We call these generically “tunnel keys.” Given that one needs to use a tunnel key at all, there are at least two reasonable ways to assign their values:

- Per **Logical_Switch+Physical_Locator** pair. That is, each logical switch may be assigned a different tunnel key on every **Physical_Locator**. This model is especially flexible.

In this model, **Physical_Locator** carries the tunnel key. Therefore, one **Physical_Locator** record will exist for each logical switch carried at a given IP destination.

- Per **Logical_Switch**. That is, every tunnel associated with a particular logical switch carries the same tunnel key, regardless of the **Physical_Locator** to which the tunnel is addressed. This model may ease switch implementation because it imposes fewer requirements on the hardware datapath.

In this model, **Logical_Switch** carries the tunnel key. Therefore, one **Physical_Locator** record will exist for each IP destination.

tunnel_key: optional integer

This column is used only in the tunnel key per **Logical_Switch** model (see above), because only in that model is there a tunnel key associated with a logical switch.

For **vxlan_over_ipv4** encapsulation, this column is the VXLAN VNI that identifies a logical switch. It must be in the range 0 to 16,777,215.

Identification:

name: string (must be unique within table)
Symbolic name for the logical switch.

description: string
An extended description for the logical switch, such as its switch login banner.

Ucast_Macs_Local TABLE

Mapping of unicast MAC addresses to tunnels (physical locators). This table is written by the HSC, so it contains the MAC addresses that have been learned on physical ports by a VTEP.

Summary:

MAC	string
logical_switch	Logical_Switch
locator	Physical_Locator
ipaddr	string

Details:

MAC: string
A MAC address that has been learned by the VTEP.

logical_switch: Logical_Switch
The Logical switch to which this mapping applies.

locator: Physical_Locator
The physical locator to be used to reach this MAC address. In this table, the physical locator will be one of the tunnel IP addresses of the appropriate VTEP.

ipaddr: string
The IP address to which this MAC corresponds. Optional field for the purpose of ARP supression.

Ucast_Macs_Remote TABLE

Mapping of unicast MAC addresses to tunnels (physical locators). This table is written by the NVC, so it contains the MAC addresses that the NVC has learned. These include VM MAC addresses, in which case the physical locators will be hypervisor IP addresses. The NVC will also report MACs that it has learned from other HSCs in the network, in which case the physical locators will be tunnel IP addresses of the corresponding VTEPs.

Summary:

MAC	string
logical_switch	Logical_Switch
locator	Physical_Locator
ipaddr	string

Details:

MAC: string

A MAC address that has been learned by the NSC.

logical_switch: **Logical_Switch**

The Logical switch to which this mapping applies.

locator: **Physical_Locator**

The physical locator to be used to reach this MAC address. In this table, the physical locator will be either a hypervisor IP address or a tunnel IP addresses of another VTEP.

ipaddr: string

The IP address to which this MAC corresponds. Optional field for the purpose of ARP supression.

Mcast_Macs_Local TABLE

Mapping of multicast MAC addresses to tunnels (physical locators). This table is written by the HSC, so it contains the MAC addresses that have been learned on physical ports by a VTEP. These may be learned by IGMP snooping, for example. This table also specifies how to handle unknown unicast and broadcast packets.

Summary:

MAC	string
logical_switch	Logical_Switch
locator_set	Physical_Locator_Set

Details:

MAC: string

A MAC address that has been learned by the VTEP.

The keyword **unknown-dst** is used as a special “Ethernet address” that indicates the locations to which packets in a logical switch whose destination addresses do not otherwise appear in **Ucast_Macs_Local** (for unicast addresses) or **Mcast_Macs_Local** (for multicast addresses) should be sent.

logical_switch: **Logical_Switch**

The Logical switch to which this mapping applies.

locator_set: **Physical_Locator_Set**

The physical locator set to be used to reach this MAC address. In this table, the physical locator set will be contain one or more tunnel IP addresses of the appropriate VTEP(s).

Mcast_Macs_Remote TABLE

Mapping of multicast MAC addresses to tunnels (physical locators). This table is written by the NVC, so it contains the MAC addresses that the NVC has learned. This table also specifies how to handle unknown unicast and broadcast packets.

Multicast packet replication may be handled by a service node, in which case the physical locators will be IP addresses of service nodes. If the VTEP supports replication onto multiple tunnels, then this may be used to replicate directly onto VTEP-hyperisor tunnels.

Summary:

MAC	string
logical_switch	Logical_Switch
locator_set	Physical_Locator_Set
ipaddr	string

Details:

MAC: string

A MAC address that has been learned by the NSC.

The keyword **unknown-dst** is used as a special “Ethernet address” that indicates the locations to which packets in a logical switch whose destination addresses do not otherwise appear in **Ucast_Macs_Remote** (for unicast addresses) or **Mcast_Macs_Remote** (for multicast addresses) should be sent.

logical_switch: **Logical_Switch**

The Logical switch to which this mapping applies.

locator_set: **Physical_Locator_Set**

The physical locator set to be used to reach this MAC address. In this table, the physical locator set will be either a service node IP address or a set of tunnel IP addresses of hypervisors (and potentially other VTEPs).

ipaddr: string

The IP address to which this MAC corresponds. Optional field for the purpose of ARP supression.

Logical_Router TABLE

A logical router, or VRF. A logical router may be connected to one or more logical switches. Subnet addresses and interface addresses may be configured on the interfaces.

Summary:

switch_binding	map of string- Logical_Switch pairs
static_routes	map of string-string pairs
<i>Identification:</i>	
name	string (must be unique within table)
description	string

Details:

switch_binding: map of string-**Logical_Switch** pairs

Maps from an IPv4 or IPv6 address prefix in CIDR notation to a logical switch. Multiple prefixes may map to the same switch. By writing a 32-bit (or 128-bit for v6) address with a /N prefix length, both the router's interface address and the subnet prefix can be configured. For example, 192.68.1.1/24 creates a /24 subnet for the logical switch attached to the interface and assigns the address 192.68.1.1 to the router interface.

static_routes: map of string-string pairs

One or more static routes, mapping IP prefixes to next hop IP addresses.

Identification:

name: string (must be unique within table)

Symbolic name for the logical router.

description: string

An extended description for the logical router.

Physical_Locator_Set TABLE

A set of one or more **Physical_Locators**.

This table exists only because OVSDB does not have a way to express the type “map from string to one or more **Physical_Locator** records.”

Summary:

locators

immutable set of 1 or more **Physical_Locators**

Details:

locators: immutable set of 1 or more **Physical_Locators**

Physical Locator TABLE

Identifies an endpoint to which logical switch traffic may be encapsulated and forwarded.

For the **vxlan_over_ipv4** encapsulation, the only encapsulation defined so far, all endpoints associated with a given **Logical_Switch** must use a common tunnel key, which is carried in the **tunnel_key** column of **Logical_Switch**.

For some encapsulations yet to be defined, we expect **Physical_Locator** to identify both an endpoint and a tunnel key. When the first such encapsulation is defined, we expect to add a “tunnel_key” column to **Physical_Locator** to allow the tunnel key to be defined.

See the “Per Logical-Switch Tunnel Key” section in the **Logical_Switch** table for further discussion of the model.

Summary:

encapsulation_type	immutable string, must be vxlan_over_ipv4
dst_ip	immutable string
<i>Bidirectional Forwarding Detection (BFD):</i>	
bfd : min_rx	optional string
bfd : min_tx	optional string
bfd : cpath_down	optional string
bfd_status : state	optional string
bfd_status : forwarding	optional string
bfd_status : diagnostic	optional string
bfd_status : remote state	optional string
bfd_status : remote diagnostic	optional string

Details:

encapsulation_type: immutable string, must be **vxlan_over_ipv4**

The type of tunneling encapsulation.

dst_ip: immutable string

For **vxlan_over_ipv4** encapsulation, the IPv4 address of the VXLAN tunnel endpoint.

We expect that this column could be used for IPv4 or IPv6 addresses in encapsulations to be introduced later.

Bidirectional Forwarding Detection (BFD):

BFD, defined in RFC 5880, allows point to point detection of connectivity failures by occasional transmission of BFD control messages. It is implemented in Open vSwitch to serve as a more popular and standards compliant alternative to CFM.

BFD operates by regularly transmitting BFD control messages at a rate negotiated independently in each direction. Each endpoint specifies the rate at which it expects to receive control messages, and the rate at which it's willing to transmit them. An endpoint which fails to receive BFD control messages for a period of three times the expected reception rate, will signal a connectivity fault. In the case of a unidirectional connectivity issue, the system not receiving BFD control messages will signal the problem to its peer in the messages it transmits.

The Open vSwitch implementation of BFD aims to comply faithfully with the requirements put forth in RFC 5880. Currently, the only known omission is "Demand Mode", which we hope to include in future. Open vSwitch does not implement the optional Authentication or "Echo Mode" features.

bfd : min_rx: optional string

The minimum rate, in milliseconds, at which this BFD session is willing to receive BFD control messages. The actual rate may be slower if the remote endpoint isn't willing to transmit as quickly as specified. Defaults to **1000**.

bfd : min_tx: optional string

The minimum rate, in milliseconds, at which this BFD session is willing to transmit BFD control messages. The actual rate may be slower if the remote endpoint isn't willing to receive as quickly as specified. Defaults to **100**.

bfd : cpath_down: optional string

Concatenated path down may be used when the local system should not have traffic forwarded to it for some reason other than a connectivity failure on the interface being monitored. The local BFD session will notify the remote session of the connectivity problem, at which time the remote session may choose not to forward traffic. Defaults to **false**.

bfd_status : state: optional string

State of the BFD session. One of **ADMIN_DOWN**, **DOWN**, **INIT**, or **UP**.

bfd_status : forwarding: optional string

True if the BFD session believes this **Physical_Locator** may be used to forward traffic. Typically this means the local session is up, and the remote system isn't signalling a problem such as concatenated path down.

bfd_status : diagnostic: optional string

A short message indicating what the BFD session thinks is wrong in case of a problem.

bfd_status : remote state: optional string

State of the remote endpoint's BFD session.

bfd_status : remote diagnostic: optional string

A short message indicating what the remote endpoint's BFD session thinks is wrong in case of a problem.