NAME

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 NSX.

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. NSX.
VRF	Virtual Routing and Forwarding instance.

Common Column

Some tables contain a column, named **other_config**. This column has the same form and purpose each place that it appears, so we describe it here to save space later.

other_config: map of string-string pairs
Key-value pairs for configuring rarely used or proprietary features.

Some tables do not have **other_config** column because no key-value pairs have yet been defined for them.

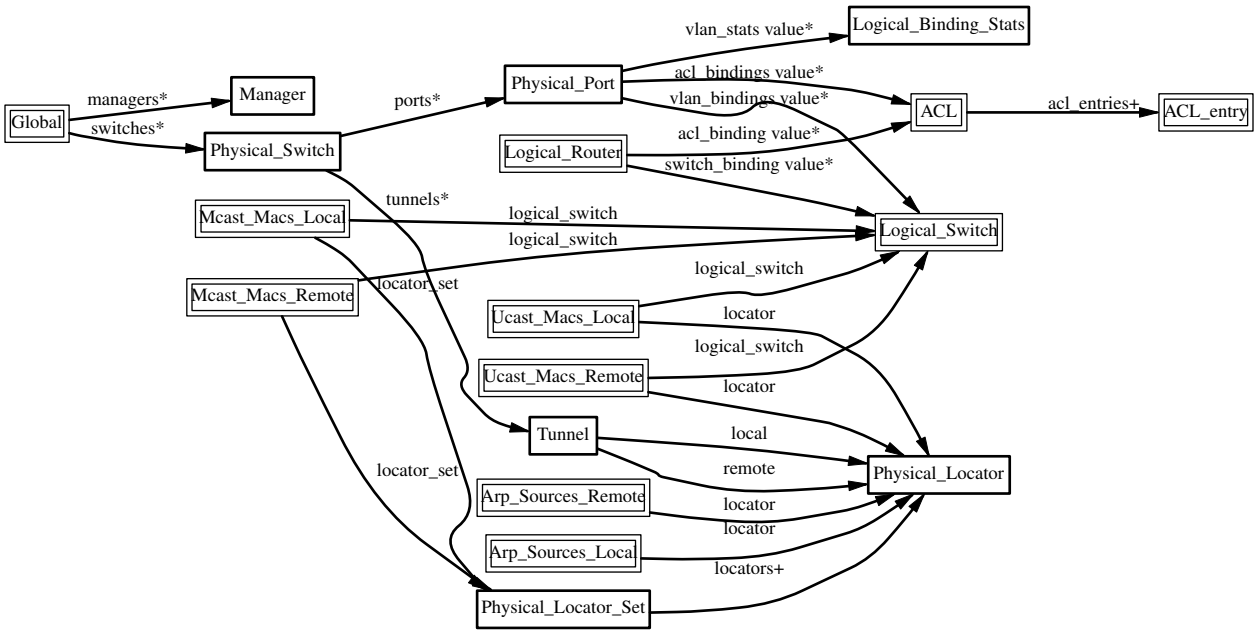
TABLE SUMMARY

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.
Tunnel	A tunnel created by 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.
Arp_Sources_Local	ARP source addresses for logical routers
Arp_Sources_Remote	ARP source addresses for logical routers
Physical_Locator_Set	Physical_Locator_Set configuration.
Physical_Locator	Physical_Locator configuration.
ACL_entry	ACL_entry configuration.
ACL	ACL 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_Switch**s
Database Configuration:	
managers	set of **Manager**s
Common Column:	
other_config	map of string-string pairs

Details:

switches: set of **Physical_Switch**s

The physical switch or switches managed by the VTEP.

When a physical switch integrates support for this VTEP schema, which is expected to be the most common case, this column should point to one **Physical_Switch** record that represents the switch itself. In another possible implementation, a server or a VM presents a VTEP schema front-end interface to one or more physical switches, presumably communicating with those physical switches over a proprietary protocol. In that case, this column would point to one **Physical_Switch** for each physical switch, and the set might change over time as the front-end server comes to represent a differing set of switches.

Database Configuration:

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

managers: set of **Manager**s

Database clients to which the 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.

Common Column:

The overall purpose of this column is described under **Common Column** at the beginning of this document.

other_config: map of string-string pairs

Manager TABLE

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

The 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**, **BACKOFF**, **CONNECTING**, **IDLE**, or **VOID**
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:*host*[**:***port*]

The specified SSL *port* (default: 6640) on the given *host*, which can either be a DNS name (if built with unbound library) or an IP address.

SSL key and certificate configuration happens outside the database.

tcp:*host*[**:***port*]

The specified TCP *port* (default: 6640) on the given *host*, which can either be a DNS name (if built with unbound library) or an IP address.

pssl:[*port*][**:***host*]

Listens for SSL connections on the specified TCP *port* (default: 6640). If *host*, which can either be a DNS name (if built with unbound library) or an IP address, is specified, then connections are restricted to the resolved or specified local IP address.

ptcp:[*port*][**:***host*]

Listens for connections on the specified TCP *port* (default: 6640). If *host*, which can either be a DNS name (if built with unbound library) or an IP address, is specified, then connections are restricted to the resolved or 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 the Open vSwitch database 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, the database server 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**, **BACKOFF**, **CONNECTING**, **IDLE**, or **VOID**
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 database server.

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 database server. 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_Port**s
tunnels	set of **Tunnel**s
Network Status:	
management_ips	set of strings
tunnel_ips	set of strings
Identification:	
name	string (must be unique within table)
description	string
Error Notification:	
switch_fault_status : mac_table_exhaustion	none
switch_fault_status : tunnel_exhaustion	none
switch_fault_status : lr_switch_bindings_fault	none
switch_fault_status : lr_static_routes_fault	none
switch_fault_status : lr_creation_fault	none
switch_fault_status : lr_support_fault	none
switch_fault_status : unspecified_fault	none
switch_fault_status : unsupported_source_node_replication	none
Common Column:	
other_config	map of string-string pairs

Details:

ports: set of **Physical_Port**s

The physical ports within the switch.

tunnels: set of **Tunnel**s

Tunnels created by this switch as instructed by the NVC.

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.

Error Notification:

An entry in this column indicates to the NVC that this switch has encountered a fault. The switch must clear this column when the fault has been cleared.

switch_fault_status : mac_table_exhaustion: none

Indicates that the switch has been unable to process MAC entries requested by the NVC due to lack of table resources.

switch_fault_status : tunnel_exhaustion: none
Indicates that the switch has been unable to create tunnels requested by the NVC due to lack of resources.

switch_fault_status : lr_switch_bindings_fault: none
Indicates that the switch has been unable to create the logical router interfaces requested by the NVC due to conflicting configurations or a lack of hardware resources.

switch_fault_status : lr_static_routes_fault: none
Indicates that the switch has been unable to create the static routes requested by the NVC due to conflicting configurations or a lack of hardware resources.

switch_fault_status : lr_creation_fault: none
Indicates that the switch has been unable to create the logical router requested by the NVC due to conflicting configurations or a lack of hardware resources.

switch_fault_status : lr_support_fault: none
Indicates that the switch does not support logical routing.

switch_fault_status : unspecified_fault: none
Indicates that an error has occurred in the switch but that no more specific information is available.

switch_fault_status : unsupported_source_node_replication: none
Indicates that the requested source node replication mode cannot be supported by the physical switch; this specifically means in this context that the physical switch lacks the capability to support source node replication mode. This error occurs when a controller attempts to set source node replication mode for one of the logical switches that the physical switch is keeping context for. An NVC that observes this error should take appropriate action (for example reverting the logical switch to service node replication mode). It is recommended that an NVC be proactive and test for support of source node replication by using a test logical switch on vtep physical switch nodes and then trying to change the replication mode to source node on this logical switch, checking for error. The NVC could remember this capability per vtep physical switch. Using mixed replication modes on a given logical switch is not recommended. Service node replication mode is considered a basic requirement since it only requires sending a packet to a single transport node, hence it is not expected that a switch should report that service node mode cannot be supported.

Common Column:

The overall purpose of this column is described under **Common Column** at the beginning of this document.

other_config: map of string-string pairs

Tunnel TABLE

A tunnel created by a **Physical_Switch**.

Summary:

local	**Physical_Locator**
remote	**Physical_Locator**
Bidirectional Forwarding Detection (BFD):	
BFD Local Configuration:	
bfd_config_local : bfd_dst_mac	optional string
bfd_config_local : bfd_dst_ip	optional string
BFD Remote Configuration:	
bfd_config_remote : bfd_dst_mac	optional string
bfd_config_remote : bfd_dst_ip	optional string
BFD Parameters:	
bfd_params : enable	optional string, either **true** or **false**
bfd_params : min_rx	optional string, containing an integer, at least 1
bfd_params : min_tx	optional string, containing an integer, at least 1
bfd_params : decay_min_rx	optional string, containing an integer
bfd_params : forwarding_if_rx	optional string, either **true** or **false**
bfd_params : cpath_down	optional string, either **true** or **false**
bfd_params : check_tnl_key	optional string, either **true** or **false**
BFD Status:	
bfd_status : enabled	optional string, either **true** or **false**
bfd_status : state	optional string, one of **admin_down**, **down**, **init**, or **up**
bfd_status : forwarding	optional string, either **true** or **false**
bfd_status : diagnostic	optional string
bfd_status : remote_state	optional string, one of **admin_down**, **down**, **init**, or **up**
bfd_status : remote_diagnostic	optional string
bfd_status : info	optional string

Details:

local: **Physical_Locator**

Tunnel end-point local to the physical switch.

remote: **Physical_Locator**

Tunnel end-point remote to the physical switch.

Bidirectional Forwarding Detection (BFD):

BFD, defined in RFC 5880, allows point to point detection of connectivity failures by occasional transmission of BFD control messages. VTEPs are expected to implement BFD.

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.

A hardware VTEP is expected to use BFD to determine reachability of devices at the end of the tunnels with which it exchanges data. This can enable the VTEP to choose a functioning service node among a set of service nodes providing high availability. It also enables the NVC to report the health status of tunnels.

In many cases the BFD peer of a hardware VTEP will be an Open vSwitch instance. The Open vSwitch implementation of BFD aims to comply faithfully with the requirements put forth in RFC 5880. Open vSwitch does not implement the optional Authentication or ''Echo Mode'' features.

BFD Local Configuration:

The HSC writes the key-value pairs in the **bfd_config_local** column to specify the local configurations to be used for BFD sessions on this tunnel.

bfd_config_local : bfd_dst_mac: optional string
Set to an Ethernet address in the form *xx*:*xx*:*xx*:*xx*:*xx*:*xx* to set the MAC expected as destination for received BFD packets. The default is **00:23:20:00:00:01**.

bfd_config_local : bfd_dst_ip: optional string
Set to an IPv4 address to set the IP address that is expected as destination for received BFD packets. The default is **169.254.1.0**.

BFD Remote Configuration:

The **bfd_config_remote** column is the remote counterpart of the **bfd_config_local** column. The NVC writes the key-value pairs in this column.

bfd_config_remote : bfd_dst_mac: optional string
Set to an Ethernet address in the form *xx*:*xx*:*xx*:*xx*:*xx*:*xx* to set the destination MAC to be used for transmitted BFD packets. The default is **00:23:20:00:00:01**.

bfd_config_remote : bfd_dst_ip: optional string
Set to an IPv4 address to set the IP address used as destination for transmitted BFD packets. The default is **169.254.1.1**.

BFD Parameters:

The NVC sets up key-value pairs in the **bfd_params** column to enable and configure BFD.

bfd_params : enable: optional string, either **true** or **false**
True to enable BFD on this **Tunnel**. If not specified, BFD will not be enabled by default.

bfd_params : min_rx: optional string, containing an integer, at least 1
The shortest interval, in milliseconds, at which this BFD session offers to receive BFD control messages. The remote endpoint may choose to send messages at a slower rate. Defaults to **1000**.

bfd_params : min_tx: optional string, containing an integer, at least 1
The shortest interval, in milliseconds, at which this BFD session is willing to transmit BFD control messages. Messages will actually be transmitted at a slower rate if the remote endpoint is not willing to receive as quickly as specified. Defaults to **100**.

bfd_params : decay_min_rx: optional string, containing an integer
An alternate receive interval, in milliseconds, that must be greater than or equal to **bfd_params:min_rx**. The implementation should switch from **bfd_params:min_rx** to **bfd_params:decay_min_rx** when there is no obvious incoming data traffic at the tunnel, to reduce the CPU and bandwidth cost of monitoring an idle tunnel. This feature may be disabled by setting a value of 0. This feature is reset whenever **bfd_params:decay_min_rx** or **bfd_params:min_rx** changes.

bfd_params : forwarding_if_rx: optional string, either **true** or **false**
When **true**, traffic received on the **Tunnel** is used to indicate the capability of packet I/O. BFD control packets are still transmitted and received. At least one BFD control packet must be received every 100 * **bfd_params:min_rx** amount of time. Otherwise, even if traffic is received, the **bfd_params:forwarding** will be **false**.

bfd_params : cpath_down: optional string, either **true** or **false**
Set to true to notify the remote endpoint that traffic should not be forwarded to this system for some reason other than a connectivity failure on the interface being monitored. The typical underlying reason is ‘‘concatenated path down,’’ that is, that connectivity beyond the local system is down. Defaults to false.

bfd_params : check_tnl_key: optional string, either **true** or **false**
Set to true to make BFD accept only control messages with a tunnel key of zero. By default, BFD accepts control messages with any tunnel key.

BFD Status:

The VTEP sets key-value pairs in the **bfd_status** column to report the status of BFD on this tunnel. When BFD is not enabled, with **bfd_params:enable**, the HSC clears all key-value pairs from **bfd_status**.

bfd_status : enabled: optional string, either **true** or **false**
Set to true if the BFD session has been successfully enabled. Set to false if the VTEP cannot support BFD or has insufficient resources to enable BFD on this tunnel. The NVC will disable the BFD monitoring on the other side of the tunnel once this value is set to false.

bfd_status : state: optional string, one of **admin_down**, **down**, **init**, or **up**
Reports the state of the BFD session. The BFD session is fully healthy and negotiated if **UP**.

bfd_status : forwarding: optional string, either **true** or **false**
Reports whether the BFD session believes this **Tunnel** may be used to forward traffic. Typically this means the local session is signaling **UP**, and the remote system isn't signaling a problem such as concatenated path down.

bfd_status : diagnostic: optional string
A diagnostic code specifying the local system's reason for the last change in session state. The error messages are defined in section 4.1 of [RFC 5880].

bfd_status : remote_state: optional string, one of **admin_down**, **down**, **init**, or **up**
Reports the state of the remote endpoint's BFD session.

bfd_status : remote_diagnostic: optional string
A diagnostic code specifying the remote system's reason for the last change in session state. The error messages are defined in section 4.1 of [RFC 5880].

bfd_status : info: optional string
A short message providing further information about the BFD status (possibly including reasons why BFD could not be enabled).

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
acl_bindings	map of integer-**ACL** 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
Error Notification:	
port_fault_status : invalid_vlan_map	none
port_fault_status : invalid_ACL_binding	none
port_fault_status : unspecified_fault	none
Common Column:	
other_config	map of string-string pairs

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. A value of zero in the VLAN field means that untagged traffic on the physical port is mapped to the logical switch.

acl_bindings: map of integer-**ACL** pairs, key in range 0 to 4,095

Attach Access Control Lists (ACLs) to the physical port. The column consists of a map of VLAN tags to **ACL**s. If the value of the VLAN tag in the map is 0, this means that the ACL is associated with the entire physical port. Non-zero values mean that the ACL is to be applied only on packets carrying that VLAN tag value. Switches will not necessarily support matching on the VLAN tag for all ACLs, and unsupported ACL bindings will cause errors to be reported. The binding of an ACL to a specific VLAN and the binding of an ACL to the entire physical port should not be combined on a single physical port. That is, a mix of zero and non-zero keys in the map is not recommended.

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. A value of zero in the VLAN field refers to untagged traffic on the physical port.

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.

Error Notification:

An entry in this column indicates to the NVC that the physical port has encountered a fault. The switch must clear this column when the error has been cleared.

port_fault_status : invalid_vlan_map: none

Indicates that a VLAN-to-logical-switch mapping requested by the controller could not be instantiated by the switch because of a conflict with local configuration.

port_fault_status : invalid_ACL_binding: none
Indicates that an error has occurred in associating an ACL with a port.

port_fault_status : unspecified_fault: none
Indicates that an error has occurred on the port but that no more specific information is available.

Common Column:

The overall purpose of this column is described under **Common Column** at the beginning of this document.

other_config: map of string-string pairs

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
Replication Mode:	
replication_mode	optional string, either **service_node** or **source_node**
Identification:	
name	string (must be unique within table)
description	string
Common Column:	
other_config	map of string-string pairs

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, when the tunnel key per **Logical_Switch** model is in use, this column is the VXLAN VNI that identifies a logical switch. It must be in the range 0 to 16,777,215.

Replication Mode:

For handling L2 broadcast, multicast and unknown unicast traffic, packets can be sent to all members of a logical switch referenced by a physical switch. There are different modes to replicate the packets. The default mode of replication is to send the traffic to a service node, which can be a hypervisor, server or appliance, and let the service node handle replication to other transport nodes (hypervisors or other VTEP physical switches). This mode is called service node replication. An alternate mode of replication, called source node replication involves the source node sending to all other transport nodes. Hypervisors are always responsible for doing their own replication for locally attached VMs in both modes. Service node replication mode is the default and considered a basic requirement because it only requires sending the packet to a single transport node.

replication_mode: optional string, either **service_node** or **source_node**

This optional column defines the replication mode per **Logical_Switch**. There are 2 valid values, **service_node** and **source_node**. If the column is not set, the replication mode defaults to service_node.

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.

Common Column:

The overall purpose of this column is described under **Common Column** at the beginning of this document.

other_config: map of string-string pairs

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 NVC.

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**
ipaddr	string

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).

ipaddr: string

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

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, using source node replication, then this may be used to replicate directly onto VTEP-hypervisor or VTEP-VTEP 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 NVC.

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 set of service nodes when service node replication is used or the set of transport nodes (defined as hypervisors or VTEPs) participating in the associated logical switch, when source node replication is used. When service node replication is used, the VTEP should send packets to one member of the locator set that is known to be healthy and reachable, which could be determined by BFD. When source node replication is used, the VTEP should send packets to all members of the locator set.

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
acl_binding	map of string-**ACL** pairs
Identification:	
name	string (must be unique within table)
description	string
Error Notification:	
LR_fault_status : invalid_ACL_binding	none
LR_fault_status : unspecified_fault	none
Common Column:	
other_config	map of string-string pairs

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.

acl_binding: map of string-**ACL** pairs

Maps ACLs to logical router interfaces. The router interfaces are indicated using IP address notation, and must be the same interfaces created in the **switch_binding** column. For example, an ACL could be associated with the logical router interface with an address of 192.68.1.1 as defined in the example above.

Identification:

name: string (must be unique within table)

Symbolic name for the logical router.

description: string

An extended description for the logical router.

Error Notification:

An entry in this column indicates to the NVC that the HSC has encountered a fault in configuring state related to the logical router.

LR_fault_status : invalid_ACL_binding: none

Indicates that an error has occurred in associating an ACL with a logical router port.

LR_fault_status : unspecified_fault: none

Indicates that an error has occurred in configuring the logical router but that no more specific information is available.

Common Column:

The overall purpose of this column is described under **Common Column** at the beginning of this document.

other_config: map of string-string pairs

Arp_Sources_Local TABLE

MAC address to be used when a VTEP issues ARP requests on behalf of a logical router.

A distributed logical router is implemented by a set of VTEPs (both hardware VTEPs and vswitches). In order for a given VTEP to populate the local ARP cache for a logical router, it issues ARP requests with a source MAC address that is unique to the VTEP. A single per-VTEP MAC can be re-used across all logical networks. This table contains the MACs that are used by the VTEPs of a given HSC. The table provides the mapping from MAC to physical locator for each VTEP so that replies to the ARP requests can be sent back to the correct VTEP using the appropriate physical locator.

Summary:

src_mac	string
locator	**Physical_Locator**

Details:

src_mac: string

The source MAC to be used by a given VTEP.

locator: **Physical_Locator**

The **Physical_Locator** to use for replies to ARP requests from this MAC address.

Arp_Sources_Remote TABLE

MAC address to be used when a remote VTEP issues ARP requests on behalf of a logical router.

This table is the remote counterpart of **Arp_sources_local**. The NVC writes this table to notify the HSC of the MACs that will be used by remote VTEPs when they issue ARP requests on behalf of a distributed logical router.

Summary:

src_mac	string
locator	**Physical_Locator**

Details:

src_mac: string

The source MAC to be used by a given VTEP.

locator: **Physical_Locator**

The **Physical_Locator** to use for replies to ARP requests from this MAC address.

Physical_Locator_Set TABLE

A set of one or more **Physical_Locator**s.

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_Locator**s

Details:

locators: immutable set of 1 or more **Physical_Locator**s

Physical_Locator TABLE

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

The **vxlan_over_ipv4** encapsulation, the only encapsulation defined so far, can use either tunnel key model described in the ‘‘Per Logical-Switch Tunnel Key’’ section in the **Logical_Switch** table. When the tunnel key per **Logical_Switch** model is in use, the **tunnel_key** column in the **Logical_Switch** table is filled with a VNI and the **tunnel_key** column in this table is empty; in the key-per-tunnel model, the opposite is true. The former model is older, and thus likely to be more widely supported. 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
tunnel_key	optional integer

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.

tunnel_key: optional integer

This column is used only in the tunnel key per **Logical_Switch+Physical_Locator** model (see above).

For **vxlan_over_ipv4** encapsulation, when the **Logical_Switch+Physical_Locator** model is in use, this column is the VXLAN VNI. It must be in the range 0 to 16,777,215.

ACL_entry TABLE

Describes the individual entries that comprise an Access Control List.

Each entry in the table is a single rule to match on certain header fields. While there are a large number of fields that can be matched on, most hardware cannot match on arbitrary combinations of fields. It is common to match on either L2 fields (described below in the L2 group of columns) or L3/L4 fields (the L3/L4 group of columns) but not both. The hardware switch controller may log an error if an ACL entry requires it to match on an incompatible mixture of fields.

Summary:

sequence	integer
L2 fields:	
source_mac	optional string
dest_mac	optional string
ethertype	optional string
L3/L4 fields:	
source_ip	optional string
source_mask	optional string
dest_ip	optional string
dest_mask	optional string
protocol	optional integer
source_port_min	optional integer
source_port_max	optional integer
dest_port_min	optional integer
dest_port_max	optional integer
tcp_flags	optional integer
tcp_flags_mask	optional integer
icmp_type	optional integer
icmp_code	optional integer
direction	string, either **egress** or **ingress**
action	string, either **deny** or **permit**
Error Notification:	
acle_fault_status : invalid_acl_entry	none
acle_fault_status : unspecified_fault	none

Details:

sequence: integer

The sequence number for the ACL entry for the purpose of ordering entries in an ACL. Lower numbered entries are matched before higher numbered entries.

L2 fields:

source_mac: optional string

Source MAC address, in the form *xx*:*xx*:*xx*:*xx*:*xx*:*xx*

dest_mac: optional string

Destination MAC address, in the form *xx*:*xx*:*xx*:*xx*:*xx*:*xx*

ethertype: optional string

Ethertype in hexadecimal, in the form *0xAAAA*

L3/L4 fields:

source_ip: optional string

Source IP address, in the form *xx.xx.xx.xx* for IPv4 or appropriate colon-separated hexadecimal notation for IPv6.

source_mask: optional string

Mask that determines which bits of source_ip to match on, in the form *xx.xx.xx.xx* for IPv4 or appropriate colon-separated hexadecimal notation for IPv6.

dest_ip: optional string
Destination IP address, in the form *xx.xx.xx.xx* for IPv4 or appropriate colon-separated hexadecimal notation for IPv6.

dest_mask: optional string
Mask that determines which bits of dest_ip to match on, in the form *xx.xx.xx.xx* for IPv4 or appropriate colon-separated hexadecimal notation for IPv6.

protocol: optional integer
Protocol number in the IPv4 header, or value of the "next header" field in the IPv6 header.

source_port_min: optional integer
Lower end of the range of source port values. The value specified is included in the range.

source_port_max: optional integer
Upper end of the range of source port values. The value specified is included in the range.

dest_port_min: optional integer
Lower end of the range of destination port values. The value specified is included in the range.

dest_port_max: optional integer
Upper end of the range of destination port values. The value specified is included in the range.

tcp_flags: optional integer
Integer representing the value of TCP flags to match. For example, the SYN flag is the second least significant bit in the TCP flags. Hence a value of 2 would indicate that the "SYN" flag should be set (assuming an appropriate mask).

tcp_flags_mask: optional integer
Integer representing the mask to apply when matching TCP flags. For example, a value of 2 would imply that the "SYN" flag should be matched and all other flags ignored.

icmp_type: optional integer
ICMP type to be matched.

icmp_code: optional integer
ICMP code to be matched.

direction: string, either **egress** or **ingress**
Direction of traffic to match on the specified port, either "ingress" (toward the logical switch or router) or "egress" (leaving the logical switch or router).

action: string, either **deny** or **permit**
Action to take for this rule, either "permit" or "deny".

Error Notification:

An entry in this column indicates to the NVC that the ACL could not be configured as requested. The switch must clear this column when the error has been cleared.

acle_fault_status : invalid_acl_entry: none
Indicates that an ACL entry requested by the controller could not be instantiated by the switch, e.g. because it requires an unsupported combination of fields to be matched.

acle_fault_status : unspecified_fault: none
Indicates that an error has occurred in configuring the ACL entry but no more specific information is available.

ACL TABLE

Access Control List table. Each ACL is constructed as a set of entries from the **ACL_entry** table. Packets that are not matched by any entry in the ACL are allowed by default.

Summary:

acl_entries	set of 1 or more **ACL_entry**s
acl_name	string (must be unique within table)
Error Notification:	
acl_fault_status : invalid_acl	none
acl_fault_status : resource_shortage	none
acl_fault_status : unspecified_fault	none

Details:

acl_entries: set of 1 or more **ACL_entry**s

A set of references to entries in the **ACL_entry** table.

acl_name: string (must be unique within table)

A human readable name for the ACL, which may (for example) be displayed on the switch CLI.

Error Notification:

An entry in this column indicates to the NVC that the ACL could not be configured as requested. The switch must clear this column when the error has been cleared.

acl_fault_status : invalid_acl: none

Indicates that an ACL requested by the controller could not be instantiated by the switch, e.g., because it requires an unsupported combination of fields to be matched.

acl_fault_status : resource_shortage: none

Indicates that an ACL requested by the controller could not be instantiated by the switch due to a shortage of resources (e.g. TCAM space).

acl_fault_status : unspecified_fault: none

Indicates that an error has occurred in configuring the ACL but no more specific information is available.