NAME

ovn-ic-sb − OVN_IC_Southbound database schema

This database holds configuration and state for interconnecting different OVN deployments. The content of the database is populated and used by the **ovn−ic** program in each OVN deployment, and not supposed to be directly used by CMS or end user.

The OVN Interconnection Southbound database is shared by **ovn−ic** program in each OVN deployment. It contains interconnection information from all related OVN deployments, and is used as the intermediate store for each OVN deployment to exchange the information. The **ovn−ic** program in each deployment is responsible for syncing the data between this database and the its own northbound and southbound databases.

Database Structure

The OVN Interconnection Southbound database contains classes of data with different properties, as described in the sections below.

Availability Zone Specific Information

These tables contain objects that are availability zone specific. Each object is owned and populated by one availability zone, and read by other availability zones.

The **Availability_Zone**, **Gateway**, **Encap** and **Port_Binding** tables are the availability zone specific tables.

Global Information

The data that does not belong to any specific availability zone but is common for all availability zones.

The **Datapath_Binding** table contains the common datapath binding information.

Common Columns

Each of the tables in this database contains a special column, named **external_ids**. This column has the same form and purpose each place it appears.

external_ids: map of string-string pairs
Key-value pairs for use by **ovn−ic**.

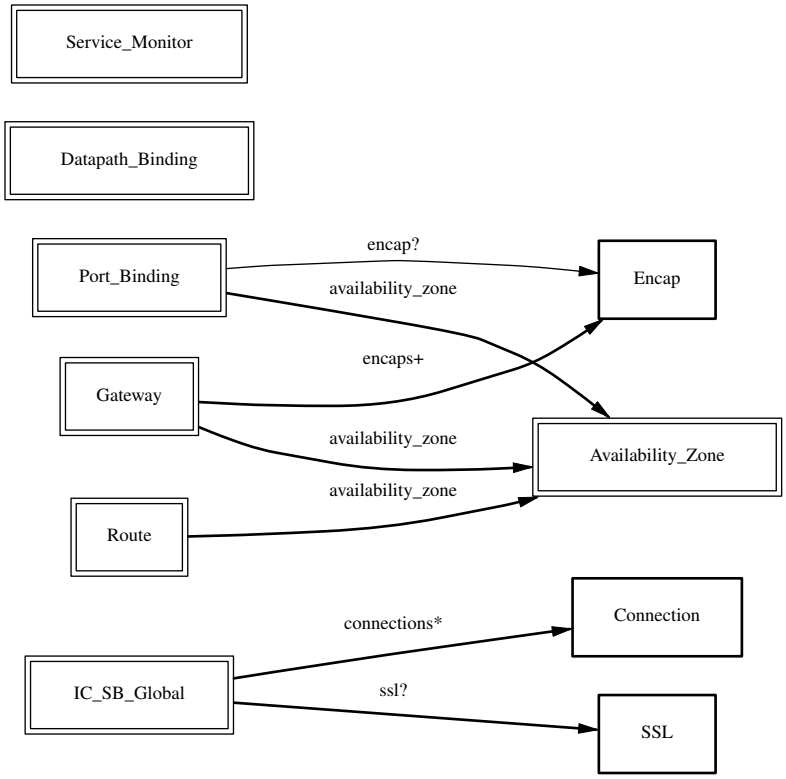
TABLE SUMMARY

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

Table	Purpose
IC_SB_Global	IC Southbound configuration
Availability_Zone	Availability Zone Information
Gateway	Interconnection Gateway Information
Encap	Encapsulation Types
Datapath_Binding	Transit Switch Datapath Bindings
Port_Binding	Transit Port Bindings
Route	Route
Connection	OVSDB client connections.
SSL	SSL configuration.
Service_Monitor	Service_Monitor 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.



IC_SB_Global TABLE

Interconnection Southbound configuration. This table must have exactly one row.

Summary:

Status:	
nb_ic_cfg	integer
Common Columns:	
external_ids	map of string-string pairs
options	map of string-string pairs
Connection Options:	
connections	set of **Connection**s
ssl	optional **SSL**

Details:

Status:

This column allow a client to track the overall configuration state of the system.

nb_ic_cfg: integer

Sequence number for the configuration. When a CMS or **ovn−ic−nbctl** updates the Interconnect northbound database, it increments the **nb_ic_cfg** column in the **NB_IC_Global** table in the Interconnect northbound database. when **OVN−ICs** updates the southbound database to bring it up to date with these changes, one **OVN−IC** updates this column to the same value.

Common Columns:

external_ids: map of string-string pairs

See **External IDs** at the beginning of this document.

options: map of string-string pairs

Connection Options:

connections: set of **Connection**s

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

ssl: optional **SSL**

Global SSL/TLS configuration.

Availability_Zone TABLE

Each row in this table represents an Availability Zone. Each OVN deployment is considered an availability zone from OVN control plane perspective, with its own central components, such as northbound and southbound databases and **ovn−northd** daemon.

Summary:

name	string (must be unique within table)
nb_ic_cfg	integer

Details:

name: string (must be unique within table)

A name that uniquely identifies the availability zone.

nb_ic_cfg: integer

This column is used by the **OVN−IC** to inform that this IC instance is aligned with the changes in INB

Gateway TABLE

Each row in this table represents a interconnection gateway chassis in an availability zone.

Summary:

name	string (must be unique within table)
availability_zone	**Availability_Zone**
hostname	string
Common Columns:	
external_ids	map of string-string pairs
Encapsulation Configuration:	
encaps	set of 1 or more **Encap**s

Details:

name: string (must be unique within table)

The name of the gateway. See **name** column of the OVN Southbound database's **Chassis** table.

availability_zone: **Availability_Zone**

The availability zone that the gateway belongs to.

hostname: string

The hostname of the gateway.

Common Columns:

The overall purpose of these columns is described under **Common Columns** at the beginning of this document.

external_ids: map of string-string pairs

Encapsulation Configuration:

OVN uses encapsulation to transmit logical dataplane packets between gateways.

encaps: set of 1 or more **Encap**s

Points to supported encapsulation configurations to transmit logical dataplane packets to this gateway. Each entry is a **Encap** record that describes the configuration. See **encaps** column of the OVN Southbound database's **Chassis** table.

Encap TABLE

The **encaps** column in the **Gateway** table refers to rows in this table to identify how OVN may transmit logical dataplane packets to this gateway.

Summary:

type	string, either **geneve** or **vxlan**
options	map of string-string pairs
ip	string
gateway_name	string

Details:

type: string, either **geneve** or **vxlan**

The encapsulation to use to transmit packets to this gateway. See **type** column of the OVN Southbound database's **Encap** table.

options: map of string-string pairs

Options for configuring the encapsulation, which may be **type** specific. See **options** column of the OVN Southbound database's **Encap** table.

ip: string

The IPv4 address of the encapsulation tunnel endpoint.

gateway_name: string

The name of the gateway that created this encap.

Datapath_Binding TABLE

Each row in this table represents a logical datapath for a transit logical switch configured in the OVN Interconnection Northbound database's **Transit_Switch** table.

Summary:

transit_switch	string
type	optional string, either **transit−router** or **transit−switch**
nb_ic_uuid	optional uuid
tunnel_key	integer, in range 1 to 16,777,215
Common Columns:	
external_ids	map of string-string pairs

Details:

transit_switch: string

The name of the transit logical switch or router that is configured in the OVN Interconnection Northbound database's **Transit_Switch** or **Transit_Router** table. Note: The name of transit router is stored in the same column due to the backward compatibility.

type: optional string, either **transit−router** or **transit−switch**

Can be one of **transit−switch** or **transit−router**

nb_ic_uuid: optional uuid

This is the UUID of the corresponding northbound IC logical datapath represented by this southbound **Datapath_Binding**.

tunnel_key: integer, in range 1 to 16,777,215

The tunnel key value to which the logical datapath is bound. The key can be generated by any **ovn−ic** but the same key is shared by all availability zones so that the logical datapaths can be peered across them. A tunnel key for transit switch datapath binding must be globally unique.

For more information about the meanings of a tunnel key, see **tunnel_key** column of the OVN Southbound database's **Datapath_Binding** table.

Common Columns:

The overall purpose of these columns is described under **Common Columns** at the beginning of this document.

external_ids: map of string-string pairs

Port_Binding TABLE

Each row in this table binds a logical port on the transit switch to a physical gateway and a tunnel key. Each port on the transit switch belongs to a specific availability zone.

Summary:

Core Features:

transit_switch	string
logical_port	string (must be unique within table)
availability_zone	**Availability_Zone**
encap	optional weak reference to **Encap**
gateway	string
tunnel_key	integer, in range 1 to 32,767
address	string
type	optional string, either **transit−router−port** or **transit−switch−port**
nb_ic_uuid	optional uuid

Common Columns:

external_ids	map of string-string pairs

Details:

Core Features:

transit_switch: string

The name of the transit switch that the corresponding logical port belongs to.

logical_port: string (must be unique within table)

A logical port, taken from **name** in the OVN_Northbound database's **Logical_Switch_Port** table. The logical port name must be unique across all availability zones.

availability_zone: **Availability_Zone**

The availability zone that the port belongs to.

encap: optional weak reference to **Encap**

Points to supported encapsulation configurations to transmit logical dataplane packets to this gateway. Each entry is a **Encap** record that describes the configuration.

gateway: string

The name of the gateway that this port is physically located.

tunnel_key: integer, in range 1 to 32,767

A number that represents the logical port in the key (e.g. Geneve TLV) field carried within tunnel protocol packets. The key can be generated by any **ovn−ic** but the same key is shared by all availability zones so that the packets can go through the datapath pipelines of different availability zones.

The tunnel ID must be unique within the scope of a logical datapath.

For more information about tunnel key, see **tunnel_key** column of the OVN Southbound database's **Port_Binding** table.

address: string

The Ethernet address and IP addresses used by the corresponding logical router port peering with the transit switch port. It is a string combined with the value of **mac** column followed by the values in **networks** column in **Logical_Router_Port** table.

type: optional string, either **transit−router−port** or **transit−switch−port**

Can be one of **switch−port** or **router−port**

nb_ic_uuid: optional uuid

This is the UUID of the corresponding northbound IC logical datapath represented by this southbound **Datapath_Binding**.

Common Columns:

external_ids: map of string-string pairs
See **External IDs** at the beginning of this document.

Route TABLE

Each row in this table represents a route advertised.

Summary:

Core Features:

transit_switch	string
availability_zone	**Availability_Zone**
route_table	string
ip_prefix	string
nexthop	string
origin	string, one of **connected**, **loadbalancer**, or **static**

Common Columns:

external_ids	map of string-string pairs

Details:

Core Features:

transit_switch: string
The name of the transit switch, upon which the route is advertised.

availability_zone: **Availability_Zone**
The availability zone that has advertised the route.

route_table: string
Route table within which this route was created. Empty value means *<main>* routing table.

Routes for directly-connected networks will be learned to *<main>* routing table and if Logical Routers have more than one Transit Switch, which interconnects them, directly-connected routes will be added via each transit switch port and configured as ECMP routes.

Static routes within route tables will be advertised and learned only if interconnecting transit switch's LRPs will have same value in **options:route_table** as NB **route_table** or ICSB **route_table** value respectively.

ip_prefix: string
IP prefix of this route (e.g. 192.168.100.0/24).

nexthop: string
Nexthop IP address for this route.

origin: string, one of **connected**, **loadbalancer**, or **static**
Can be one of **connected**, **static** or **loadbalancer**. Routes to directly-connected subnets - LRP's CIDRs are inserted to OVN IC SB DB with **connected** value in **origin**. Static routes are inserted to OVN IC SB DB with **static** value. Routes for LB VIPs are inserted in OVN IC SB DB with **loadbalancer** value. Next when route is learned to another AZ NB DB by ovn-ic, route origin is synced to **options:origin**.

Common Columns:

external_ids: map of string-string pairs
See **External IDs** at the beginning of this document.

Connection TABLE

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

This table primarily configures the Open vSwitch database server (**ovsdb−server**).

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**, **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
status : bound_port	optional string, containing an integer
Common Columns:	
external_ids	map of string-string pairs
other_config	map of string-string pairs

Details:

Core Features:

target: string (must be unique within table)

Connection methods for clients.

The following connection methods are currently supported:

ssl:*host*[**:***port*]

The specified SSL/TLS *port* on the given *host*, which can either be a DNS name (if built with unbound library) or an IP address. A valid SSL/TLS configuration must be provided when this form is used, this configuration can be specified via command-line options or the **SSL** table.

If *port* is not specified, it defaults to 6640.

SSL/TLS support is an optional feature that is not always built as part of OVN or Open vSwitch.

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

The specified TCP *port* on the given *host*, which can either be a DNS name (if built with unbound library) or an IP address (IPv4 or IPv6). If *host* is an IPv6 address, wrap it in square brackets, e.g. **tcp:[::1]:6640**.

If *port* is not specified, it defaults to 6640.

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

Listens for SSL/TLS connections on the specified TCP *port*. Specify 0 for *port* to have the kernel automatically choose an available port. 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 (either IPv4 or IPv6 address). If *host* is an IPv6 address, wrap in square brackets, e.g. **pssl:6640:[::1]**. If *host* is not

specified then it listens only on IPv4 (but not IPv6) addresses. A valid SSL/TLS configuration must be provided when this form is used, this can be specified either via command-line options or the **SSL** table.

If *port* is not specified, it defaults to 6640.

SSL/TLS support is an optional feature that is not always built as part of OVN or Open vSwitch.

ptcp:[*port*][**:***host*]
Listens for connections on the specified TCP *port*. Specify 0 for *port* to have the kernel automatically choose an available port. 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 (either IPv4 or IPv6 address). If *host* is an IPv6 address, wrap it in square brackets, e.g. **ptcp:6640:[::1]**. If *host* is not specified then it listens only on IPv4 addresses.

If *port* is not specified, it defaults to 6640.

When multiple clients are configured, the **target** values must be unique. Duplicate **target** values yield unspecified results.

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:

Key-value pair of **is_connected** is always updated. Other key-value pairs in the status columns may be updated depends on the **target** type.

When **target** specifies a connection method that listens for inbound connections (e.g. **ptcp:** or **punix:**), both **n_connections** and **is_connected** may also be updated while the remaining key-value pairs are omitted.

On the other hand, when **target** specifies an outbound connection, all key-value pairs may be updated, except the above-mentioned two key-value pairs associated with inbound connection targets. They are omitted.

is_connected: boolean
true if currently connected to this client, **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 client last successfully connected to the database (in seconds). Value is empty if client has never successfully been connected.

status : sec_since_disconnect: optional string, containing an integer, at least 0
The amount of time since this client last disconnected from the database (in seconds). Value is empty if client 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.

status : bound_port: optional string, containing an integer
When **target** is **ptcp:** or **pssl:**, this is the TCP port on which the OVSDB server is listening. (This is particularly useful when **target** specifies a port of 0, allowing the kernel to choose any available port.)

Common Columns:

The overall purpose of these columns is described under **Common Columns** at the beginning of this document.

external_ids: map of string-string pairs

other_config: map of string-string pairs

SSL TABLE

SSL/TLS configuration for ovn-sb database access.

Summary:

private_key	string
certificate	string
ca_cert	string
bootstrap_ca_cert	boolean
ssl_protocols	string
ssl_ciphers	string
ssl_ciphersuites	string
Common Columns:	
external_ids	map of string-string pairs

Details:

private_key: string

Name of a PEM file containing the private key used as the switch's identity for SSL/TLS connections to the controller.

certificate: string

Name of a PEM file containing a certificate, signed by the certificate authority (CA) used by the controller and manager, that certifies the switch's private key, identifying a trustworthy switch.

ca_cert: string

Name of a PEM file containing the CA certificate used to verify that the switch is connected to a trustworthy controller.

bootstrap_ca_cert: boolean

If set to **true**, then Open vSwitch will attempt to obtain the CA certificate from the controller on its first SSL/TLS connection and save it to the named PEM file. If it is successful, it will immediately drop the connection and reconnect, and from then on all SSL/TLS connections must be authenticated by a certificate signed by the CA certificate thus obtained. **This option exposes the SSL/TLS connection to a man−in−the−middle attack obtaining the initial CA certificate.** It may still be useful for bootstrapping.

ssl_protocols: string

Range or a comma- or space-delimited list of the SSL/TLS protocols to enable for SSL/TLS connections.

Supported protocols include **TLSv1.2** and **TLSv1.3**. Ranges can be provided in a form of two protocol names separated with a dash (**TLSv1.2−TLSv1.3**), or as a single protocol name with a plus sign (**TLSv1.2+**). The value can be a list of protocols or exactly one range. The range is a preferred way of specifying protocols and the configuration always behaves as if the range between the minimum and the maximum specified version is provided, i.e., if the value is set to **TLSv1.X,TLSv1.(X+2)**, the **TLSv1.(X+1)** will also be enabled as if it was a range. Regardless of order, the highest protocol supported by both sides will be chosen when making the connection.

The default when this option is omitted is **TLSv1.2+**.

ssl_ciphers: string

List of ciphers (in OpenSSL cipher string format) to be supported for SSL/TLS connections with TLSv1.2. The default when this option is omitted is **DEFAULT:@SECLEVEL=2**.

ssl_ciphersuites: string

List of ciphersuites (in OpenSSL ciphersuites string format) to be supported for SSL/TLS connections with TLSv1.3 and later. Default value from OpenSSL will be used when this option is omitted.

Common Columns:

The overall purpose of these columns is described under **Common Columns** at the beginning of this document.

external_ids: map of string-string pairs

Service_Monitor TABLE

Summary:

type	optional string, must be **load−balancer**
ip	string
protocol	optional string, either **tcp** or **udp**
port	integer, in range 0 to 65,535
logical_port	string
src_ip	string
src_mac	string
status	optional string, one of **error**, **offline**, or **online**
target_availability_zone	string
source_availability_zone	string
options	map of string-string pairs
external_ids	map of string-string pairs

Details:

type: optional string, must be **load−balancer**
: The type of the service. Only the value "load-balancer" is supported.

ip: string
: IP of the service to be monitored. Copy from SBDB record.

protocol: optional string, either **tcp** or **udp**
: The protocol of the service. Copy from source Southbound Database record.

port: integer, in range 0 to 65,535
: The TCP or UDP port of the service. Copy from source Southbound Database record.

logical_port: string
: The VIF of the logical port on which the service is running. The **ovn−controller** that binds this **logical_port** monitors the service by sending periodic monitor packets. Copy from source Southbound Database record.

src_ip: string
: Source IPv4 or IPv6 address to use in the service monitor packet. Copy from source Southbound Database record.

src_mac: string
: Source Ethernet address to use in the service monitor packet. Copy from Southbound Database record.

status: optional string, one of **error**, **offline**, or **online**
: The health status of the service monitor, synchronized from target Southbound Database.

target_availability_zone: string
: The Availability Zone where the monitored service endpoint is located.

source_availability_zone: string
: Availability Zone that initiated this monitor entry in ICSB and retains ownership for lifecycle management.

options: map of string-string pairs
: Same as in Table Service_Monitor in SBDB. Copy from source SBDB record.

external_ids: map of string-string pairs
: Same as in Table Service_Monitor in SBDB. Copy from source SBDB record.