

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **Subport interface , .1D Bridges , bridge port interface** |
| **Authors** | **Mellanox** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **07/18/2016** |
| **SAI-Version** | **1.0** |

1 Overview 3

2 Sub port interface 3

2.1 New port type attribute 3

2.2 Example - Router sub port flow 4

3 Non vlan aware bridge domains (.1D Bridges) 4

3.1 Non Vlan aware bridge flows 5

4 Bridge ports 6

4.1 SAI bridge/router object current state 6

4.2 SAI bridge/router object proposed state 7

4.3 Bridge port type types 7

4.3.1 Bridge port type - Port 8

4.3.2 Bridge vport type – { Port , vlan} 8

4.3.3 Bridge port type- router 8

4.3.4 Bridge port type – tunnel 9

4.4 Bridge attributes 10

5 Specification 10

5.1 SAI port object 10

5.2 SAI FDB object 10

5.3 SAI STP object 11

5.4 SAI VLAN member object 11

5.5 SAI router interface object 11

5.1 SAI tunnel 12

5.2 SAI bridge port object 13

5.2.1 Bridge port API summary 15

5.3 SAI bridge object 15

5.3.1 Bridge API summary 16

6 Examples 16

6.1 1.D bridge with router example 16

6.2 1.Q bridge with router example 19

6.3 1.D bridge with VXlan example 20

7 References 22

# Overview

The proposal contain three prats

* In order build a discrete pipeline and well define layers additional set of object are needed - Bridge ports. Currently when entering the switch we have -> ingress port /LAG, when entering a router we have SAI RIF. A bridge port object is missing.
* Increase SAI 4k vlan broadcast domain by adding multiple vlan unaware bridges (.1D) to the existing Vlan aware bridge (1.Q)
* Add an ability to define an interface base on {port,vlan} were interface can be router interface or bridge interface

# Sub port interface

Ability to define an interface base on {port,vlan} were interface can be router interface or bridge interface



## New port type attribute

Control on the port mode optional modes

* Port – legacy
* Sub port- port is split into logical interfaces, each interface is defined by {port,vlan}

## Example - Router sub port flow



# Non vlan aware bridge domains (.1D Bridges)

Increase SAI 4k vlan broadcast domain by adding multiple vlan unaware bridges (.1D) to the existing vlan aware bridge (1.Q)

Add ability to create Non vlan aware bridge object

Add ability to bind an "interface" to the a .1D bridge

* only interface type {port,vlan}

Please note that since PVID assignment is done prior to the interface classification untagged packets

can be classified based on {port,PVID}



## Non Vlan aware bridge flows



# Bridge ports

In order build a discrete pipeline and well defined layers, additional set of object are needed - Bridge ports. Currently when entering the switch we have -> ingress port /LAG, when entering a router we have SAI RIF. A bridge port object is missing.

## SAI bridge/router object current state



## SAI bridge/router object proposed state



## Bridge port type types

Port – represents phy port or LAG traditional .1Q bridge port

Vport - represents phy port or LAG vlan .1D bridge port interface

Router - represents the port that connects the bridge to the router

Tunnel - represents the port that connects the bridge to tunnel



.

### Bridge port type - Port

Represents phy port or LAG traditional .1Q bridge port

Can be added into .1Q Bridge only

Can be binded only form a port mode {port}

By default all ports will have a corresponding bride port added to the default 1.q bridge

Valid attribute:



### Bridge vport type – { Port , vlan}

Represents phy port or LAG .vlan

Can be added into .1D Bridge only

Can be binded only form a port mode {vport}



### Bridge port type- router

Represents the bridge port connection to the router

Two types of router bridge port

.1Q router bridge port- a single port representing all router Vlan interfaces

Per bridge .1D router bridge port per .1D bridge port



### Bridge port type – tunnel

Represents the bridge port connection to a tunnel

There is one to one mapping between the bride port and the tunnel



## Bridge attributes

All bridge related attribute will move from port/LAG to bridge port object

Bridge attribute

STP

Vlan objects

…

# Specification

## SAI port object

**typedef enum \_sai\_port\_bind\_mode\_attr\_t**

**{**

**SAI\_PORT\_BIND\_MODE\_PORT,**

**SAI\_PORT\_BIND\_MODE\_SUB\_PORT,**

**} sai\_port\_bind\_mode\_attr\_t ;**

typedef enum \_sai\_port\_attr\_t

{

//new attribute

/\*\* port type[sai\_port\_bind\_attr\_t] \*/

**SAI\_PORT\_BIND\_MODE,**

//removed atrtributes

SAI\_PORT\_ATTR\_FDB\_LEARNING,

SAI\_PORT\_ATTR\_MAX\_LEARNED\_ADDRESSES,

SAI\_PORT\_ATTR\_FDB\_LEARNING\_LIMIT\_VIOLATION ,

} sai\_port\_attr\_t ;

## SAI FDB object

**typedef enum \_ sai\_fdb\_entry\_bridge\_type\_t**

**{**

**SAI\_FDB\_ENTRY\_BRIDGE\_TYPE\_1Q,**

**SAI\_FDB\_ENTRY\_BRIDGE\_TYPE\_1D,**

**} sai\_fdb\_entry\_bridge\_type\_t ;**

typedef struct \_sai\_fdb\_entry\_t

{

sai\_mac\_t mac\_address;

**sai\_fdb\_entry\_br\_type br\_type; // .1D or .1Q**

**//valid for .1Q**

**sai\_vlan\_id\_t vlan\_id;**

**//valid for .1D**

**sai\_bridge\_id\_t br\_id;**

} sai\_fdb\_entry\_t;

typedef enum \_sai\_fdb\_entry\_attr\_t

{

//new attribute

SAI\_FDB\_ENTRY\_ATTR\_BRIDGE\_PORT\_ID

//removed attributes

SAI\_FDB\_ENTRY\_ATTR\_PORT\_ID

}

## SAI STP object

Set set port state- Panding STP port state – change to object

The port state should be move from port to bridge port

typedef enum \_sai\_stp\_attr\_t

{

SAI\_STP\_ATTR\_START,

/\*\* READ-ONLY \*/

/\*\* Vlans attached to STP instance [sai\_vlan\_list\_t] \*/

SAI\_STP\_ATTR\_VLAN\_LIST = SAI\_STP\_ATTR\_START,

//new attribute

/\*\* bridge attached to STP instance \*/

SAI\_STP\_ATTR\_BRIDGE\_ID

/\*\* READ-WRITE \*/

SAI\_STP\_ATTR\_END,

} sai\_stp\_attr\_t;

## SAI VLAN member object

typedef enum \_sai\_vlan\_member\_attr\_t {

SAI\_VLAN\_MEMBER\_ATTR\_START,

/\*\* READ\_WRITE \*/

/\*\* VLAN ID [sai\_vlan\_id\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_VLAN\_MEMBER\_ATTR\_VLAN\_ID = SAI\_VLAN\_MEMBER\_ATTR\_START,

/\*\* logical port ID [sai\_object\_id\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_VLAN\_MEMBER\_ATTR\_PORT\_ID,

**/\*\* logical bridge port ID valid only for .1Q bridge ports [sai\_object\_id\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/**

**SAI\_VLAN\_MEMBER\_ATTR\_BRIDGE\_PORT\_ID,**

/\*\* VLAN tagging mode [sai\_vlan\_tagging\_mode\_t] (CREATE\_AND\_SET)

\* (default to SAI\_VLAN\_PORT\_UNTAGGED) \*/

SAI\_VLAN\_MEMBER\_ATTR\_TAGGING\_MODE,

SAI\_VLAN\_MEMBER\_ATTR\_END,

/\*\* custom range base value \*/

SAI\_VLAN\_MEMBER\_ATTR\_CUSTOM\_RANGE\_START = 0x10000000,

/\* --\*/

SAI\_VLAN\_MEMBER\_ATTR\_CUSTOM\_RANGE\_END

} sai\_vlan\_member\_attr\_t;

## SAI router interface object

/\*\*

\* @brief Attribute data for SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE

\*/

typedef enum \_sai\_router\_interface\_type\_t

{

/\*\* Port or Lag Router Interface Type \*/

SAI\_ROUTER\_INTERFACE\_TYPE\_PORT,

/\*\* VLAN Router Interface Type \*/

SAI\_ROUTER\_INTERFACE\_TYPE\_VLAN,

/\*\* Loopback Router Interface Type \*/

SAI\_ROUTER\_INTERFACE\_TYPE\_LOOPBACK

**/\*\* sub port Interface Type \*/**

**SAI\_ROUTER\_INTERFACE\_TYPE\_SUB\_PORT**

**/\*\* .1D bridge Interface Type \*/**

**SAI\_ROUTER\_INTERFACE\_TYPE\_BRIDGE**

} sai\_router\_interface\_type\_t;

typedef enum \_sai\_router\_interface\_attr\_t

{

/\*\* READ-ONLY \*/

/\*\* Virtual router id [sai\_object\_id\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_ROUTER\_INTERFACE\_ATTR\_VIRTUAL\_ROUTER\_ID,

/\*\* Type [sai\_router\_interface\_type\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE,

/\*\* Assosiated Port or Lag object id [sai\_object\_id\_t]

\* (MANDATORY\_ON\_CREATE when SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE == SAI\_ROUTER\_INTERFACE\_TYPE\_PORT or SUB\_PORT | CREATE\_ONLY)

\*/

SAI\_ROUTER\_INTERFACE\_ATTR\_PORT\_ID,

/\*\* Assosiated Vlan [sai\_vlan\_id\_t]

\* (MANDATORY\_ON\_CREATE when SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE == SAI\_ROUTER\_INTERFACE\_TYPE\_VLAN or SUB\_PORT | CREATE\_ONLY)

\*/

SAI\_ROUTER\_INTERFACE\_ATTR\_VLAN\_ID,

## SAI tunnel

typedef enum \_sai\_tunnel\_map\_type\_t

{

/\*\* TUNNEL Map overlay ECN to underlay ECN (only valid for encap) \*/

SAI\_TUNNEL\_MAP\_TYPE\_OECN\_TO\_UECN = 0x00000000,

/\*\* TUNNEL Map underlay ECN and overlay ECN to overlay ECN (only valid for decap) \*/

SAI\_TUNNEL\_MAP\_TYPE\_UECN\_OECN\_TO\_OECN = 0x00000001,

/\*\* TUNNEL Map VNI to VLAN ID \*/

SAI\_TUNNEL\_MAP\_TYPE\_VNI\_TO\_VLAN\_ID = 0x00000002,

/\*\* TUNNEL Map VLAN ID to VNI \*/

SAI\_TUNNEL\_MAP\_TYPE\_VLAN\_ID\_TO\_VNI = 0x00000003,

/\*\* TUNNEL Map VNI to Bridge IF \*/

SAI\_TUNNEL\_MAP\_TYPE\_VNI\_TO\_BRIDGE\_IF = 0x00000004,

/\*\* TUNNEL Map Bridge IF to VNI \*/

SAI\_TUNNEL\_MAP\_TYPE\_BRIDGE\_IF\_TO\_VNI = 0x00000005,

/\*\* Custom range base value \*/

SAI\_TUNNEL\_MAP\_TYPE\_CUSTOM\_RANGE\_BASE = 0x10000000

} sai\_tunnel\_map\_type\_t;

## SAI bridge port object

/\*\*

\* @brief Attribute data for SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE

\*/

typedef enum \_sai\_bridge\_port\_type\_t

{

/\*\* Port or Lag \*/

SAI\_BRIDGE\_PORT\_TYPE\_PORT,

/\*\* {Port or Lag.vlan} \*/

SAI\_BRIDGE\_PORT\_TYPE\_SUB\_PORT,

/\*\* bridge router port \*/

SAI\_BRIDGE\_PORT\_TYPE\_1Q\_ROUTER,

/\*\* bridge router port \*/

SAI\_BRIDGE\_PORT\_TYPE\_1D\_ROUTER,

/\*\* bridge tunnel port \*/

SAI\_BRIDGE\_PORT\_TYPE\_TUNNEL,

} sai\_bridge\_port\_type\_t;

typedef enum \_sai\_bridge\_port\_attr\_t

{

/\*\* Type [sai\_bridge\_port\_type\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_BRIDGE\_PORT\_ATTR\_TYPE,

/\*\* Assosiated Port or Lag object id [sai\_object\_id\_t]

\* (MANDATORY\_ON\_CREATE when SAI\_BRIDEG\_PORT\_ATTR\_TYPE == SAI\_BRIDGE\_PORT\_TYPE\_PORT or SAI\_BRIDGE\_PORT\_TYPE\_SUB\_PORT | CREATE\_ONLY)

\*/

SAI\_BRIDGE\_PORT\_ATTR\_PORT\_ID,

/\*\* Assosiated vlan object id [sai\_object\_id\_t]

\* (MANDATORY\_ON\_CREATE when SAI\_BRIDEG\_PORT\_ATTR\_TYPE == SAI\_BRIDGE\_PORT\_TYPE\_SUB\_PORT | CREATE\_ONLY)

\*/

SAI\_BRIDGE\_PORT\_ATTR\_VLAN\_ID,

/\*\* Assosiated rouer inerface object id [sai\_object\_id\_t]

\* (MANDATORY\_ON\_CREATE when SAI\_BRIDEG\_PORT\_ATTR\_TYPE == SAI\_BRIDGE\_PORT\_TYPE\_1D\_ROUTER | CREATE\_ONLY) please note the for bridge type SAI\_BRIDGE\_PORT\_TYPE\_1Q\_ROUTER all vlan intarfaces are auto bounded to the bridge port

\*/

SAI\_BRIDGE\_PORT\_ATTR\_RIF\_ID,

/\*\* Assosiated tunnel id [sai\_object\_id\_t]

\* (MANDATORY\_ON\_CREATE when SAI\_BRIDEG\_PORT\_ATTR\_TYPE == SAI\_BRIDGE\_PORT\_TYPE\_TUNNEL | CREATE\_ONLY)

\*/

SAI\_BRIDGE\_PORT\_ATTR\_TUNNEL\_ID,

/\*\* Assosiated bridge ID [sai\_object\_id\_t]

\* (CREATE\_AND\_SET defult invalid bridge )

\*/

SAI\_BRIDGE\_PORT\_ATTR\_BRIDGE\_ID,

//bridge port attribute defualt leraning enadle

SAI\_BRIDGE\_PORT\_ATTR\_FDB\_LEARNING,

// defualt no limit

SAI\_BRIDGE\_PORT\_ATTR\_MAX\_LEARNED\_ADDRESSES,

SAI\_BRIDGE\_PORT\_ATTR\_FDB\_LEARNING\_LIMIT\_VIOLATION ,

} sai\_bridge\_port\_attr\_t ;

/\*\*

\* @brief Create bridge port

\*

\* @param[out] bridge\_port\_id

\* @param[in] attr\_count number of attributes

\* @param[in] attr\_list array of attributes

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_create\_bridge\_port\_fn)(

\_Out\_ sai\_object\_id\_t\* bridge\_port\_id,

\_In\_ uint32\_t attr\_count,

\_In\_ const sai\_attribute\_t \*attr\_list

);

/\*\*

\* @brief Remove bridge port

\*

\* @param[in] bridge\_port\_id.

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_bridge\_port\_fn) (

\_In\_ sai\_object\_id\_t bridge\_port\_id

);

/\*\*

\* @brief Set attribute for bridge port

\*

\* @param[in] bridge port id

\* @param[in] attr attribute to set

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_bridge\_port\_attribute\_fn)(

\_In\_ sai\_object\_id\_t bridge\_port\_id,

\_In\_ const sai\_attribute\_t \*attr

);

/\*\*

\* @brief Get attributes of bridge port

\*

\* @param[in] bridge\_port\_id

\* @param[in] attr\_count number of attributes

\* @param[inout] attr\_list array of attributes

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_get\_bridge\_port\_attribute\_fn)(

\_In\_ sai\_object\_id\_t bridge\_port\_id ,

\_In\_ uint32\_t attr\_count,

\_Inout\_ sai\_attribute\_t \*attr\_list

);

### Bridge port API summary

typedef struct bridge\_port\_t

{

sai\_create\_bridge\_port\_fn create\_bridge\_port;

sai\_remove\_bridge\_port\_fn remove\_bridge\_port;

sai\_set\_bridge\_port\_attribute\_fn set\_bridge\_port\_attribute;

sai\_get\_bridge\_port\_attribute\_fn get\_bridge\_port\_attribute;

} sai\_bridge\_port\_api\_t;

## SAI bridge object

typedef enum \_sai\_bridge\_type\_t

{

SAI\_BRIDGE\_TYPE\_1Q,

SAI\_BRIDGE\_TYPE\_1D,

} sai\_bridge\_type\_t;

typedef enum \_sai\_bridge\_attr\_t

{

/\*\* SAI bridge type vlan\_aware , non\_vlan\_aware [sai\_bridge\_type\_t] \*/

SAI\_BRIDGE\_ATTR\_TYPE,

} sai\_bridge\_attr\_t ;

/\*\*

\* @brief Create bridge

\*

\* @param[out] bridge\_port\_id

\* @param[in] attr\_count number of attributes

\* @param[in] attr\_list array of attributes

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_create\_bridge\_fn)(

\_Out\_ sai\_object\_id\_t\* bridge\_id,

\_In\_ uint32\_t attr\_count,

\_In\_ const sai\_attribute\_t \*attr\_list

);

/\*\*

\* @brief Remove bridge

\*

\* @param[in] bridge\_id.

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_bridge\_fn) (

\_In\_ sai\_object\_id\_t bridge\_id

);

/\*\*

\* @brief Set attribute for bridge

\*

\* @param[in] bridge id

\* @param[in] attr attribute to set

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_bridge\_attribute\_fn)(

\_In\_ sai\_object\_id\_t bridge\_id,

\_In\_ const sai\_attribute\_t \*attr

);

/\*\*

\* @brief Get attributes of bridge

\*

\* @param[in] bridge\_id bridge\_id

\* @param[in] attr\_count number of attributes

\* @param[inout] attr\_list array of attributes

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_get\_bridge\_attribute\_fn)(

\_In\_ sai\_object\_id\_t bridge\_id ,

\_In\_ uint32\_t attr\_count,

\_Inout\_ sai\_attribute\_t \*attr\_list

);

### Bridge API summary

typedef struct bridge\_t

{

sai\_create\_bridge\_fn create\_bridge;

sai\_remove\_bridge\_fn remove\_bridge;

sai\_set\_bridge\_attribute\_fn set\_bridge\_attribute;

sai\_get\_bridge\_attribute\_fn get\_bridge\_attribute;

} sai\_bridge\_api\_t;

# Examples

## 1.D bridge with router example



//configure port 3,4,5,6 to sub port mode

sai\_attribute\_t port\_attr;

port\_attr.id=**SAI\_PORT\_BIND\_MODE**;

port\_attr.value= **SAI\_PORT\_BIND\_MODE\_SUB\_PORT**;

sai\_set\_port\_attribute\_fn(port\_3,port\_attr);

sai\_set\_port\_attribute\_fn(port\_4,port\_attr);

//create bridges

sai\_object\_id bridge1

sai\_object\_id bridge2

sai\_attribute\_t bridge\_attr\_list[1];

bridge\_attr\_list[0].id=SAI\_BRIDGE\_ATTR\_TYPE;

bridge\_attr\_list[0].value=SAI\_BRIDGE\_TYPE\_1D;

sai\_create\_bridge\_fn(&bridge1,1, bridge\_attr\_list);

sai\_create\_bridge\_fn(&bridge2,1, bridge\_attr\_list);

//create vports and add vport to bridge

sai\_object\_id vport3.7,vport3.6,vport4.16.vport4.17;

sai\_attribute\_t vport\_attr\_list[4];

vport\_attr\_list[0].id= SAI\_BRIDEG\_PORT\_ATTR\_TYPE;

vport\_attr\_list[0].value=SAI\_BRIDGE\_PORT\_TYPE\_SUB\_PORT;

vport\_attr\_list[1].id= SAI\_BRIDGE\_PORT\_ATTR\_PORT\_ID;

vport\_attr\_list[1].value=port3;

vport\_attr\_list[2].id= SAI\_BRIDGE\_PORT\_ATTR\_VLAN\_ID;

vport\_attr\_list[2].value=vlan7;

vport\_attr\_list[3].id= SAI\_BRIDGE\_PORT\_ATTR\_BRIDGE\_ID;

vport\_attr\_list[3].value=bridge1;

sai\_create\_bridge\_port\_fn(&vport3.7,4, vport\_attr\_list);

vport\_attr\_list[0].id= SAI\_BRIDEG\_PORT\_ATTR\_TYPE;

vport\_attr\_list[0].value=SAI\_BRIDGE\_PORT\_TYPE\_SUB\_PORT;

vport\_attr\_list[1].id= SAI\_BRIDGE\_PORT\_ATTR\_PORT\_ID;

vport\_attr\_list[1].value=port3;

vport\_attr\_list[2].id= SAI\_BRIDGE\_PORT\_ATTR\_VLAN\_ID;

vport\_attr\_list[2].value=vlan6;

vport\_attr\_list[3].id= SAI\_BRIDGE\_PORT\_ATTR\_BRIDGE\_ID;

vport\_attr\_list[3].value=bridge1;

sai\_create\_bridge\_port\_fn(&vport3.6,4, vport\_attr\_list);

//create rif …

sai\_object\_id bridge1\_rif, bridge2\_rif;

sai\_attribute\_t rif\_attr\_list[2];

rif\_attr\_list[0].id= SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE;

rif\_attr\_list[0].value=**SAI\_ROUTER\_INTERFACE\_TYPE\_BRIDGE**;

rif\_attr\_list[1].id= SAI\_ROUTER\_INTERFACE\_ATTR\_VIRTUAL\_ROUTER\_ID;

rif\_attr\_list[1].value=**VRF\_default**;

sai\_create\_router\_if\_fn(&bridge1\_rif,2, rif\_attr\_list);

rif\_attr\_list[0].id= SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE;

rif\_attr\_list[0].value=**SAI\_ROUTER\_INTERFACE\_TYPE\_BRIDGE**;

rif\_attr\_list[1].id= SAI\_ROUTER\_INTERFACE\_ATTR\_VIRTUAL\_ROUTER\_ID;

rif\_attr\_list[1].value=**VRF\_default**;

sai\_create\_router\_if\_fn(&bridge2\_rif,2, rif\_attr\_list);

//create bridge router port & add interface to bridge (connect the RIF to the bridge)

sai\_object\_id bridge1\_rport,bridge2\_rport;

vport\_attr\_list[0].id= SAI\_BRIDGE\_PORT\_ATTR\_TYPE;

vport\_attr\_list[0].value=SAI\_BRIDGE\_PORT\_TYPE\_1D\_ROUTER;

vport\_attr\_list[1].id= SAI\_BRIDGE\_PORT\_ATTR\_RIF\_ID;

vport\_attr\_list[1].value=bridge1\_rif;

vport\_attr\_list[2].id= SAI\_BRIDGE\_PORT\_ATTR\_BRIDGE\_ID;

vport\_attr\_list[2].value=bridge1;

sai\_create\_bridge\_port\_fn(&bridge1\_rport,3, vport\_attr\_list);

vport\_attr\_list[0].id= SAI\_BRIDGE\_PORT\_ATTR\_TYPE;

vport\_attr\_list[0].value=SAI\_BRIDGE\_PORT\_TYPE\_1D\_ROUTER;

vport\_attr\_list[1].id= SAI\_BRIDGE\_PORT\_ATTR\_RIF\_ID;

vport\_attr\_list[1].value=bridge2\_rif;

vport\_attr\_list[2].id= SAI\_BRIDGE\_PORT\_ATTR\_BRIDGE\_ID;

vport\_attr\_list[2].value=bridge1;

sai\_create\_bridge\_port\_fn(&bridge2\_rport,3, vport\_attr\_list);

## 1.Q bridge with router example

//configure port 3,4,5,6 to sub port mode

sai\_attribute\_t port\_attr;

port\_attr.id=**SAI\_PORT\_BIND\_MODE**;

port\_attr.value= **SAI\_PORT\_BIND\_MODE\_PORT**;

sai\_set\_port\_attribute\_fn(port\_3,port\_attr);

sai\_set\_port\_attribute\_fn(port\_4,port\_attr);

//create bridges

sai\_object\_id bridge1

sai\_attribute\_t bridge\_attr\_list[1];

bridge\_attr\_list[0].id=SAI\_BRIDGE\_ATTR\_TYPE;

bridge\_attr\_list[0].value=SAI\_BRIDGE\_TYPE\_1Q;

sai\_create\_bridge\_fn(&bridge1,1, bridge\_attr\_list);

//create bridge ports and add ports to bridge

sai\_object\_id bport3,bport4;

sai\_attribute\_t bport\_attr\_list[3];

vport\_attr\_list[0].id= SAI\_BRIDEG\_PORT\_ATTR\_TYPE;

vport\_attr\_list[0].value=SAI\_BRIDGE\_PORT\_TYPE\_PORT;

vport\_attr\_list[1].id= SAI\_BRIDGE\_PORT\_ATTR\_PORT\_ID;

vport\_attr\_list[1].value=port3;

vport\_attr\_list[2].id= SAI\_BRIDGE\_PORT\_ATTR\_BRIDGE\_ID;

vport\_attr\_list[2].value=bridge1;

sai\_create\_bridge\_port\_fn(&bport3,3, vport\_attr\_list);

vport\_attr\_list[0].id= SAI\_BRIDEG\_PORT\_ATTR\_TYPE;

vport\_attr\_list[0].value=SAI\_BRIDGE\_PORT\_TYPE\_PORT;

vport\_attr\_list[1].id= SAI\_BRIDGE\_PORT\_ATTR\_PORT\_ID;

vport\_attr\_list[1].value=port4;

vport\_attr\_list[2].id= SAI\_BRIDGE\_PORT\_ATTR\_BRIDGE\_ID;

vport\_attr\_list[2].value=bridge1;

sai\_create\_bridge\_port\_fn(&bport4,3, vport\_attr\_list);

//create vlan rif(s) …

//create bridge router port & add interface to bridge (connect the RIF to the bridge)

sai\_object\_id bridge1\_rport;

vport\_attr\_list[0].id= SAI\_BRIDGE\_PORT\_ATTR\_TYPE;

vport\_attr\_list[0].value=SAI\_BRIDGE\_PORT\_TYPE\_1Q\_ROUTER;

vport\_attr\_list[1].id= SAI\_BRIDGE\_PORT\_ATTR\_BRIDGE\_ID;

vport\_attr\_list[1].value=bridge1;

sai\_create\_bridge\_port\_fn(&bridge1\_rport,2, vport\_attr\_list);

The default .1Q router can be read by quering SAI\_SWITCH\_ATTR\_DEFAULT\_1Q\_BRIDGE\_ID

For every port on system init, there is instance of bridge port connecting the port to the default .1Q bridge. The list of this bridge ports can be read by querying SAI\_BRIDGE\_ATTR\_PORT\_LIST of the .1Q bridge instance.

## 1.D bridge with VXlan example

//configure port 3,4,5,6 to sub port mode

sai\_attribute\_t port\_attr;

port\_attr.id=**SAI\_PORT\_BIND\_MODE**;

port\_attr.value= **SAI\_PORT\_BIND\_MODE\_SUB\_PORT**;

sai\_set\_port\_attribute\_fn(port\_3,port\_attr);

sai\_set\_port\_attribute\_fn(port\_4,port\_attr);

//create bridges

sai\_object\_id bridge1

sai\_object\_id bridge2

sai\_attribute\_t bridge\_attr\_list[1];

bridge\_attr\_list[0].id=SAI\_BRIDGE\_ATTR\_TYPE;

bridge\_attr\_list[0].value=SAI\_BRIDGE\_TYPE\_1D;

sai\_create\_bridge\_fn(&bridge1,1, bridge\_attr\_list);

sai\_create\_bridge\_fn(&bridge2,1, bridge\_attr\_list);

//create vports and add vport to bridge

sai\_object\_id vport3.7,vport3.6,vport4.16.vport4.17;

sai\_attribute\_t vport\_attr\_list[4];

vport\_attr\_list[0].id= SAI\_BRIDEG\_PORT\_ATTR\_TYPE;

vport\_attr\_list[0].value=SAI\_BRIDGE\_PORT\_TYPE\_SUB\_PORT;

vport\_attr\_list[1].id= SAI\_BRIDGE\_PORT\_ATTR\_PORT\_ID;

vport\_attr\_list[1].value=port3;

vport\_attr\_list[2].id= SAI\_BRIDGE\_PORT\_ATTR\_VLAN\_ID;

vport\_attr\_list[2].value=vlan7;

vport\_attr\_list[3].id= SAI\_BRIDGE\_PORT\_ATTR\_BRIDGE\_ID;

vport\_attr\_list[3].value=bridge1;

sai\_create\_bridge\_port\_fn(&vport3.7,4, vport\_attr\_list);

vport\_attr\_list[0].id= SAI\_BRIDEG\_PORT\_ATTR\_TYPE;

vport\_attr\_list[0].value=SAI\_BRIDGE\_PORT\_TYPE\_SUB\_PORT;

vport\_attr\_list[1].id= SAI\_BRIDGE\_PORT\_ATTR\_PORT\_ID;

vport\_attr\_list[1].value=port3;

vport\_attr\_list[2].id= SAI\_BRIDGE\_PORT\_ATTR\_VLAN\_ID;

vport\_attr\_list[2].value=vlan6;

vport\_attr\_list[3].id= SAI\_BRIDGE\_PORT\_ATTR\_BRIDGE\_ID;

vport\_attr\_list[3].value=bridge1;

sai\_create\_bridge\_port\_fn(&vport3.6,4, vport\_attr\_list);

//create rif …

sai\_object\_id bridge1\_rif, bridge2\_rif;

sai\_attribute\_t rif\_attr\_list[2];

rif\_attr\_list[0].id= SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE;

rif\_attr\_list[0].value=**SAI\_ROUTER\_INTERFACE\_TYPE\_BRIDGE**;

rif\_attr\_list[1].id= SAI\_ROUTER\_INTERFACE\_ATTR\_VIRTUAL\_ROUTER\_ID;

rif\_attr\_list[1].value=**VRF\_default**;

sai\_create\_router\_if\_fn(&bridge1\_rif,2, rif\_attr\_list);

rif\_attr\_list[0].id= SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE;

rif\_attr\_list[0].value=**SAI\_ROUTER\_INTERFACE\_TYPE\_BRIDGE**;

rif\_attr\_list[1].id= SAI\_ROUTER\_INTERFACE\_ATTR\_VIRTUAL\_ROUTER\_ID;

rif\_attr\_list[1].value=**VRF\_default**;

sai\_create\_router\_if\_fn(&bridge2\_rif,2, rif\_attr\_list);

//create bridge router port & add interface to bridge (connect the RIF to the bridge)

sai\_object\_id bridge1\_rport,bridge2\_rport;

vport\_attr\_list[0].id= SAI\_BRIDEG\_PORT\_ATTR\_TYPE;

vport\_attr\_list[0].value=SAI\_BRIDGE\_PORT\_TYPE\_1D\_ROUTER;

vport\_attr\_list[1].id= SAI\_BRIDGE\_PORT\_ATTR\_RIF\_ID;

vport\_attr\_list[1].value=bridge1\_rif;

vport\_attr\_list[2].id= SAI\_BRIDGE\_PORT\_ATTR\_BRIDGE\_ID;

vport\_attr\_list[2].value=bridge1;

sai\_create\_bridge\_port\_fn(&bridge1\_rport,3, vport\_attr\_list);

vport\_attr\_list[0].id= SAI\_BRIDEG\_PORT\_ATTR\_TYPE;

vport\_attr\_list[0].value=SAI\_BRIDGE\_PORT\_TYPE\_1D\_ROUTER;

vport\_attr\_list[1].id= SAI\_BRIDGE\_PORT\_ATTR\_RIF\_ID;

vport\_attr\_list[1].value=bridge2\_rif;

vport\_attr\_list[2].id= SAI\_BRIDGE\_PORT\_ATTR\_BRIDGE\_ID;

vport\_attr\_list[2].value=bridge1;

sai\_create\_bridge\_port\_fn(&bridge2\_rport,3, vport\_attr\_list);

//create the Vxlan tunnel

//Create underlay VRF

sai\_object\_id VRF\_underlay;

… (VRF creataion flow )

//Create vxlan RIF

sai\_object\_id underlay\_rif;

sai\_attribute\_t rif\_attr\_list[2];

rif\_attr\_list[0].id= SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE;

rif\_attr\_list[0].value=SAI\_ROUTER\_INTERFACE\_TYPE\_LOOPBACK;

rif\_attr\_list[1].id= SAI\_ROUTER\_INTERFACE\_ATTR\_VIRTUAL\_ROUTER\_ID;

rif\_attr\_list[1].value=**VRF\_**underlay;

sai\_create\_router\_if\_fn(&vxlan\_RIF,2, rif\_attr\_list);

//create tunnel

sai\_object\_id vxlan\_tunnel;

sai\_attribute\_t tunnel\_attr\_list[4];

tunnel\_attr\_list[0].id= SAI\_TUNNEL\_ATTR\_TYPE;

tunnel\_attr\_list[0].value= SAI\_TUNNEL\_VXLAN;

tunnel\_attr\_list[1].id= SAI\_TUNNEL\_UNDERLAY\_INTERFACE;

tunnel\_attr\_list[1].value=underlay\_rif;

tunnel\_attr\_list[2].id= SAI\_TUNNEL\_IP\_VER;

tunnel\_attr\_list[2].value=ipv4;

tunnel\_attr\_list[3].id= SAI\_TUNNEL\_SRC\_IP;

tunnel\_attr\_list[3].value=1.1.1.1;

sai\_create\_tunnel\_fn(&vxlan\_tunnel,4, tunnel\_attr\_list);

//create tunnel bridge ports and add ports to bridge

sai\_object\_id bridge1\_tport,bridge2\_tport;

vport\_attr\_list[0].id= SAI\_BRIDEG\_PORT\_ATTR\_TYPE;

vport\_attr\_list[0].value=SAI\_BRIDGE\_PORT\_TYPE\_TUNNEL;

vport\_attr\_list[1].id= SAI\_BRIDGE\_PORT\_ATTR\_TUNNEL\_ID;

vport\_attr\_list[1].value= vxlan\_tunnel;

vport\_attr\_list[2].id= SAI\_BRIDGE\_PORT\_ATTR\_BRIDGE\_ID;

vport\_attr\_list[2].value=bridge1;

sai\_create\_bridge\_port\_fn(&bridge1\_tport,3, vport\_attr\_list);

vport\_attr\_list[0].id= SAI\_BRIDEG\_PORT\_ATTR\_TYPE;

vport\_attr\_list[0].value=SAI\_BRIDGE\_PORT\_TYPE\_TUNNEL;

vport\_attr\_list[1].id= SAI\_BRIDGE\_PORT\_ATTR\_TUNNEL\_ID;

vport\_attr\_list[1].value=vxlan\_tunnel;

vport\_attr\_list[2].id= SAI\_BRIDGE\_PORT\_ATTR\_BRIDGE\_ID;

vport\_attr\_list[2].value=bridge2;

sai\_create\_bridge\_port\_fn(&bridge2\_tport,3, vport\_attr\_list);

// update the encap and decap tunnel mapper

sai\_attribute\_t tunnel\_mapper\_attr;

tunnel\_mapper\_attr.id= SAI\_TUNNEL\_DECAP\_MAPPER;

tunnel\_mapper\_attr.value={type-SAI\_TUNNEL\_MAP\_TYPE\_VNI\_TO\_BRIDGE\_IF

map { vni1-> bridge1\_tport

vni2-> bridge2\_tport}

sai\_set\_tunnel\_attribute\_fn(vxlan\_tunnel,&tunnel\_mapper\_attr);

tunnel\_mapper\_attr.id= SAI\_TUNNEL\_ENCAP\_MAPPER;

tunnel\_mapper\_attr.value={type- SAI\_TUNNEL\_MAP\_TYPE\_BRIDGE\_IF\_TO\_VNI

map { bridge1\_tport->nvi1

bridge2\_tport->vni2}

sai\_set\_tunnel\_attribute\_fn(vxlan\_tunnel,&tunnel\_mapper\_attr);

# References