

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **Mirroring API’s** |
| **Authors** | **DELL** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **02/05/2015** |
| **SAI-Version** | **V0.9.2** |

**Contents**

[List of Changes i](#_Toc416273042)

[1 Overview 1](#_Toc416273043)

[2 Specification 1](#_Toc416273044)

[2.1 Changes to sai.h 1](#_Toc416273045)

[2.2 Changes to saiport.h 1](#_Toc416273046)

[2.3 New definitions in saimirror.h 1](#_Toc416273047)

[3 Examples 5](#_Toc416273048)

[3.1 Create SPAN mirror session 5](#_Toc416273049)

[3.2 Retrieve SPAN mirror session attributes 6](#_Toc416273050)

[3.3 Set SPAN mirror session attributes 6](#_Toc416273051)

[3.4 Associate Mirror session on a port for ingress mirroring 6](#_Toc416273052)

[3.5 Associate Mirror session on a port for egress mirroring 7](#_Toc416273053)

[3.6 Disassociate Mirror session on the port for ingress mirroring 7](#_Toc416273054)

[3.7 Create RSPAN session 8](#_Toc416273055)

[3.8 Associate SPAN and RSPAN Mirror sessions on the same port for egress mirroring 8](#_Toc416273056)

[3.9 Remove SPAN mirror session alone on the same port for egress mirroring (update of the object\_list scenario) 9](#_Toc416273057)

[3.10 Remove mirror session 10](#_Toc416273058)

[3.11 Create ERSPAN session 10](#_Toc416273059)

[3.12 Associate the ERSPAN to a acl rule 12](#_Toc416273060)

[3.13 Create ERSPAN session without a mandatory attribute 13](#_Toc416273061)

[4 Appendix 15](#_Toc416273062)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 0.9.2 | Proposal for Mirror – Version 1 |  | 2/5/15 |
| 0.9.2 | Version 2. Updated with the following changes   * Made the Mirror enable/Disable a port based attribute * Replaced all instances of SPAN with Mirroring * Added a new field for GRE Protocol Type |  | 2/16/15 |
| 0.9.2 | Version 3. Updated the following changes   * Removed the argument mirror\_type from create and added as an attribute. Appendix has more details. * Changed according to the unified objet id proposal |  | 3/17/15 |
| 0.9.2 | Version 4. Updated the following changes   * Changed few attributes to CREATE+SET. Appendix section has more details. * Changed the specification to be more specific. Appendix section has more details. * Removed SAI\_MIRROR\_L2\_TUNNEL from sai\_erspan\_encapsulation\_type\_t. Details are in Appendix section. |  | 3/20/15 |
| 0.9.2 | Version 5. Updated the following changes   * Changed the overview statement * Added Examples |  | 4/1/2015 |
| 0.9.2 | Version 6. Updated with review comments |  | 4/9/2015 |

License

© 2014 Microsoft Corporation, Dell Inc., Facebook, Inc, Broadcom Corporation, Intel Corporation, Mellanox Technologies Ltd.

As of September 9, 2014, the following persons or entities have made this Specification available under the Open Web Foundation Final Specification Agreement (OWFa 1.0), which is available at <http://www.openwebfoundation.org/legal/the-owf-1-0-agreements/owfa-1-0>

Microsoft Corporation, Dell Inc., Facebook, Inc, Intel Corporation, Mellanox Technologies Ltd.

You can review the signed copies of the Open Web Foundation Agreement Version 1.0 for this Specification at <http://opencompute.org/licensing/>, which may also include additional parties to those listed above.

Your use of this Specification may be subject to other third party rights. THIS SPECIFICATION IS PROVIDED "AS IS." The contributors expressly disclaim any warranties (express, implied, or otherwise), including implied warranties of merchantability, noninfringement, fitness for a particular purpose, or title, related to the Specification. The entire risk as to implementing or otherwise using the Specification is assumed by the Specification implementer and user. IN NO EVENT WILL ANY PARTY BE LIABLE TO ANY OTHER PARTY FOR LOST PROFITS OR ANY FORM OF INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION OR ITS GOVERNING AGREEMENT, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND WHETHER OR NOT THE OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE FOLLOWING IS A LIST OF MERELY REFERENCED TECHNOLOGY: Microprocessor technology, semiconductor manufacturing technology, operating system technology (including without limitation networking operating system technology), emulation technology, graphics technology, video technology, integrated circuit packaging technology and the like, compiler technologies, object oriented technology, optical/RF communications technology including chip I/O and driver technology, bus technology, memory chip technology (including, without limitation, NAND memory, NOR memory, resistive RAM (RRAM), seek scan probe (SSP) memory, nonvolatile memory (including without limitation, memory based on chalcogenide materials, phase change memory (PCM), one or more stacked layers of memory cells, embedded PCM memories, non-volatile cache memory, solid state drives, SRAM, embedded DRAM, ferro-electric memory, and polymer memory)) and/or health-related and medical technology. IMPLEMENTATION OF THESE TECHNOLOGIES MAY BE SUBJECT TO THEIR OWN LEGAL TERMS.

# Overview

The document describes covers details of the API to create, remove and update mirror object. The API allows user to create a SAI mirror object which defines the mirror destination, and then allows user to attach this mirror object to various source SAI objects, such as port, ACL, etc. on.

Current proposal supports three types of mirror destination, i.e., local, remote (RSPAN), and enhanced remote (ERSPAN). For local mirror, the switch mirror the packets to a local port. For RSPAN, the switch adds a VLAN tag to the mirrored packet as it egresses its local mirror port. For ERSPAN, the switch adds an encapsulation header as it egresses its local mirror port.

There are several ways to specify the mirroring source packets. When attaching the SAI mirror object to a SAI port, all packets that go into this port or come out of this port are mirrored. The proposal also support flow-based mirroring which allows user to use ACL to specify the packets to mirror.

# Specification

## Changes to sai.h

typedef enum \_sai\_api\_t {

SAI\_API\_MIRROR= 14, /\* sai\_mirror\_api\_t \*/

} sai\_api\_t;

## Changes to saiport.h

typedef enum \_sai\_port\_attr\_t

{

/\* READ-WRITE \*/

/\* Enable/Disable Mirror session [sai\_object\_list\_t]. (default to list with object\_count 0)

\* Enable ingress mirroring by assigning list of mirror session object id as attribute value

\* Disable ingress mirroring by assigning object\_count as 0 in objlist\*/

SAI\_PORT\_ATTR\_INGRESS\_MIRROR\_SESSION,

/\* READ-WRITE \*/

/\* Enable/Disable Mirror session [sai\_object\_list\_t].(default to list with object\_count 0)

\* Enable egress mirroring by assigning list of mirror session object id as attribute value

\* Disable egress mirroring by assigning object\_count as 0 in objlist\*/

SAI\_PORT\_ATTR\_EGRESS\_MIRROR\_SESSION,

} sai\_port\_attr\_t

## New definitions in saimirror.h

/\* Specify the type of mirroring \*/

typedef enum \_sai\_mirror\_type\_t {

/\* Local span \*/

SAI\_MIRROR\_TYPE\_LOCAL = 1,

/\* Remote span \*/

SAI\_MIRROR\_TYPE\_REMOTE,

/\* Enhanced Remote span \*/

SAI\_MIRROR\_TYPE\_ENHANCED\_REMOTE,

} **sai\_mirror\_type\_t**;

typedef enum \_sai\_erspan\_encapsulation\_type\_t

{

/\* L3 GRE Tunnel Encapsulation

| L2 Ethernet header | IP header | GRE header | Original mirrored packet |

\*/

SAI\_MIRROR\_L3\_GRE\_TUNNEL,

} **sai\_erspan\_encapsulation\_type\_t**;

typedef enum \_sai\_mirror\_session\_attr\_t {

/\* READ-ONLY \*/

/\* READ-WRITE \*/

/\* MANDATORY\_ON\_CREATE | CREATE \_ONLY \*/

/\* Mirror type SPAN/RSPAN/ERSPAN [sai\_mirror\_type\_t]\*/

SAI\_MIRROR\_SESSION\_ATTR\_TYPE,

/\* MANDATORY\_ON\_CREATE | CREATE\_AND\_SET\*/

/\* Destination/Analyser/Monitor Port [sai\_object\_id\_t]\*/

SAI\_MIRROR\_SESSION\_ATTR\_MONITOR\_PORT,

/\* CREATE\_AND\_SET\*/

/\* Class-of-Service (Traffic Class) [uint8\_t] (default to 0) \*/

SAI\_MIRROR\_SESSION\_ATTR\_TC,

/\* MANDATORY\_ON\_CREATE | CREATE\_AND\_SET\*/

/\* Valid for RSPAN and ERSPAN

\* L2 header TPID if vlanId is not zero [uint16\_t]\*/

SAI\_MIRROR\_SESSION\_ATTR\_VLAN\_TPID,

/\* MANDATORY\_ON\_CREATE | CREATE\_AND\_SET\*/

/\* Valid for RSPAN and ERSPAN L2 header VlanId [sai\_vlan\_id\_t]\*/

SAI\_MIRROR\_SESSION\_ATTR\_VLAN\_ID,

/\* MANDATORY\_ON\_CREATE | CREATE\_AND\_SET\*/

/\* Valid for RSPAN and ERSPAN packet priority [uint8\_t] \*/

SAI\_MIRROR\_SESSION\_ATTR\_VLAN\_PRI,

/\* All attributes below are Valid only for ERSPAN \*/

/\* MANDATORY\_ON\_CREATE | CREATE\_ONLY \*/

/\* Encapsulation type - sai\_erspan\_encapsulation\_type\_t \*/

SAI\_MIRROR\_SESSION\_ATTR\_ENCAP\_TYPE,

/\* MANDATORY\_ON\_CREATE | CREATE\_AND\_SET\*/

/\* tunnel IP header version [uint8\_t]\*/

SAI\_MIRROR\_SESSION\_ATTR\_IPHDR\_VERSION,

/\* MANDATORY\_ON\_CREATE | CREATE\_AND\_SET\*/

/\* tunnel header TOS [uint8\_t]\*/

SAI\_MIRROR\_SESSION\_ATTR\_TOS,

/\* CREATE\_AND\_SET\*/

/\* tunnel header TTL [uint8\_t] (default 255) \*/

SAI\_MIRROR\_SESSION\_ATTR\_TTL,

/\* MANDATORY\_ON\_CREATE | CREATE\_AND\_SET \*/

/\* tunnel source IP [sai\_ip\_address\_t] \*/

SAI\_MIRROR\_SESSION\_ATTR\_SRC\_IP\_ADDRESS,

/\* MANDATORY\_ON\_CREATE | CREATE\_AND\_SET \*/

/\* tunnel destination IP [sai\_ip\_address\_t] \*/

SAI\_MIRROR\_SESSION\_ATTR\_DST\_IP\_ADDRESS,

/\* MANDATORY\_ON\_CREATE | CREATE\_AND\_SET \*/

/\* L2 source MAC address [sai\_mac\_t] \*/

SAI\_MIRROR\_SESSION\_ATTR\_SRC\_MAC\_ADDRESS,

/\* MANDATORY\_ON\_CREATE | CREATE\_AND\_SET \*/

/\* L2 destination MAC address [sai\_mac\_t] \*/

SAI\_MIRROR\_SESSION\_ATTR\_DST\_MAC\_ADDRESS,

/\* MANDATORY\_ON\_CREATE | CREATE\_AND\_SET \*/

/\* GRE protocol Id [uint16\_t] \*/

SAI\_MIRROR\_SESSION\_ATTR\_GRE\_PROTOCOL\_TYPE,

} **sai\_mirror\_session\_attr\_t**;

/\*

\* Routine Description:

\* Create mirror session.

\*

\* Arguments:

\* [out] session\_id - port mirror session

\* [in] attr\_count - number of attributes

\* [in] attr\_list - array of attributes

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_create\_mirror\_session\_fn)(

\_out\_ sai\_object\_id\_t \*session\_id,

\_In\_ uint32\_t attr\_count,

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

/\*

\* Routine Description:

\* Destroy Mirror session.

\*

\* Arguments:

\* [in] session\_id - port mirror session

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_mirror\_session\_fn)(

\_In\_ sai\_object\_id\_t session\_id);

/\*

\* Routine Description:

\* Set Mirror session attribute.

\*

\* Arguments:

\* [in] session\_id - session\_id to set the attributes

\* [in] attr – attribute value

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_mirror\_session\_attribute\_fn)(

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

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

/\*

\* Routine Description:

\* Get mirroring session attribute.

\*

\* Arguments:

\* [in] session\_id - session\_id to retrieve the attribute

\* [in] attr\_count - number of attributes

\* [inout] attr\_list - array of attributes

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_get\_mirror\_session\_attribute\_fn)(

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

\_In\_ uint32\_t attr\_count,

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

/\* MIRROR method table retrieved with sai\_api\_query() \*/

typedef struct \_sai\_mirror\_api\_t

{

sai\_create\_mirror\_session\_fn create\_mirror\_session;

sai\_delete\_mirror\_session\_fn delete\_mirror\_session;

sai\_set\_mirror\_session\_attribute\_fn set\_mirror\_session\_attribute;

sai\_get\_mirror\_session\_attribute\_fn get\_mirror\_session\_attribute;

} sai\_mirror\_api\_t;

# Examples

## Create SPAN mirror session

sai\_api\_query(SAI\_API\_MIRROR, &sai\_mirror\_api);

sai\_object\_id\_t mirror\_span\_object;

sai\_attribute\_t attr[3] = {0};

attr[0].id = SAI\_MIRROR\_SESSION\_ATTR\_TC;

attr[0].value.u8 = 7;

attr[1].id = SAI\_MIRROR\_SESSION\_ATTR\_MONITOR\_PORT;

attr[1].value.oid = 104; //Embed in sai\_object\_id port numbering format

attr[2].id = SAI\_MIRROR\_SESSION\_ATTR\_TYPE;

attr[2].value.s32 = SAI\_MIRROR\_TYPE\_LOCAL;

sai\_mirror\_api->create\_mirror\_session (&mirror\_span\_object, 3,

attr);

## Retrieve SPAN mirror session attributes

sai\_attribute\_t get\_attr[3] = {0};

get\_attr[0].id = SAI\_MIRROR\_SESSION\_ATTR\_TC;

get\_attr[1].id = SAI\_MIRROR\_SESSION\_ATTR\_MONITOR\_PORT;

get\_attr[2].value.s32 = SAI\_MIRROR\_TYPE\_LOCAL;

sai\_mirror\_api->get\_mirror\_session\_attribute (mirror\_span\_object, 3, get\_attr);

Check if get\_attr[0].value.u8 is 7

Check if get\_attr[1].value.oid is 1

Check if get\_attr[2].value.s32 is SAI\_MIRROR\_TYPE\_LOCAL

## Set SPAN mirror session attributes

sai\_attribute\_t set\_attr;

sai\_attribute\_t get\_attr;

set\_attr.id = SAI\_MIRROR\_SESSION\_ATTR\_TC;

set\_attr.value.u8 = 5;

sai\_mirror\_api->set\_mirror\_session\_attribute (mirror\_span\_object, &set\_attr);

get\_attr.id = SAI\_MIRROR\_SESSION\_ATTR\_TC;

sai\_mirror\_api->get\_mirror\_session\_attribute (mirror\_span\_object, 1, &get\_attr);

Check if get\_attr[0].value.u8 is 5

## Associate Mirror session on a port for ingress mirroring

sai\_api\_query(SAI\_API\_PORT, &sai\_port\_api);

sai\_object\_id\_t port\_object\_id = 1 //Embed the port in the port numbering format.

sai\_attribute\_t set\_attr;

sai\_attribute\_t get\_attr;

set\_attr.id = SAI\_PORT\_ATTR\_INGRESS\_MIRROR\_SESSION;

set\_attr.value.objlist.object\_count = 1;

sai\_object\_id\_t \*objects = calloc (set\_attr.value.objlist.object\_count, sizeof(sai\_object\_list\_t));

objects[0] = mirror\_span\_object;

set\_attr.value.objlist.object\_list = objects;

sai\_port\_api-> set\_port\_attribute(port\_object\_id, &set\_attr);

get\_attr.id = SAI\_PORT\_ATTR\_INGRESS\_MIRROR\_SESSION;

get\_attr.value.objlist.object\_count = 1;

get\_attr.value.objlist.object\_list = calloc (get\_attr.value.objlist.object\_count, sizeof(sai\_object\_list\_t));

sai\_port\_api-> get\_port\_attribute(port\_object\_id, 1, &get\_attr);

//Check if get\_attr.value.objlist.object\_list[0] is mirror\_span\_object

## Associate Mirror session on a port for egress mirroring

sai\_api\_query(SAI\_API\_PORT, &sai\_port\_api);

sai\_object\_id\_t port\_object\_id = 1 //Embed the port in the port numbering format.

sai\_attribute\_t set\_attr;

sai\_attribute\_t get\_attr;

set\_attr.id = SAI\_PORT\_ATTR\_EGRESS\_MIRROR\_SESSION;

set\_attr.value.objlist.object\_count = 1;

sai\_object\_id\_t \*objects = calloc (set\_attr.value.objlist.object\_count, sizeof(sai\_object\_list\_t));

objects[0] = mirror\_span\_object;

set\_attr.value.objlist.object\_list = objects;

sai\_port\_api-> set\_port\_attribute(port\_object\_id, &set\_attr);

get\_attr.id = SAI\_PORT\_ATTR\_EGRESS\_MIRROR\_SESSION;

get\_attr.value.objlist.object\_count = 1;

get\_attr.value.objlist.object\_list = calloc (get\_attr.value.objlist.object\_count, sizeof(sai\_object\_list\_t));

sai\_port\_api-> get\_port\_attribute(port\_object\_id, 1, &get\_attr);

//Check if get\_attr.value.objlist.object\_list[0] is mirror\_span\_object

## Disassociate Mirror session on the port for ingress mirroring

sai\_api\_query(SAI\_API\_PORT, &sai\_port\_api);

sai\_object\_id\_t port\_object\_id = 1 //Embed the port in the port numbering format.

sai\_attribute\_t set\_attr;

set\_attr.id = SAI\_PORT\_ATTR\_INGRESS\_MIRROR\_SESSION;

set\_attr.value.objlist.object\_count = 0;

set\_attr.value.objlist.object\_list = NULL;

sai\_port\_api-> set\_port\_attribute(port\_object\_id, &set\_attr);

## Create RSPAN session

sai\_api\_query(SAI\_API\_MIRROR, &sai\_mirror\_api);

sai\_object\_id\_t mirror\_rspan\_object;

sai\_attribute\_t attr[3] = {0};

attr[0].id = SAI\_MIRROR\_SESSION\_ATTR\_TC;

attr[0].value.u8 = 7;

attr[1].id = SAI\_MIRROR\_SESSION\_ATTR\_MONITOR\_PORT;

attr[1].value.oid = 5; //Embed in sai\_object\_id port numbering format

attr[2].id = SAI\_MIRROR\_SESSION\_ATTR\_TYPE;

attr[2].value.s32 = SAI\_MIRROR\_TYPE\_REMOTE;

attr[3].id =SAI\_MIRROR\_SESSION\_ATTR\_VLAN\_TPID;

attr[3].value.u16 = 0x8100;

attr[4].id =SAI\_MIRROR\_SESSION\_ATTR\_VLAN\_ID;

attr[4].value.u16 = 0x2;

attr[5].id =SAI\_MIRROR\_SESSION\_ATTR\_VLAN\_PRI;

attr[5].value.u8 = 0x6;

sai\_mirror\_api->create\_mirror\_session (&mirror\_rspan\_object, 6,

attr);

## Associate SPAN and RSPAN Mirror sessions on the same port for egress mirroring

sai\_api\_query(SAI\_API\_PORT, &sai\_port\_api);

sai\_object\_id\_t port\_object\_id = 1 //Embed the port in the port numbering format.

sai\_attribute\_t set\_attr;

sai\_attribute\_t get\_attr;

set\_attr.id = SAI\_PORT\_ATTR\_EGRESS\_MIRROR\_SESSION;

set\_attr.value.objlist.object\_count = 2;

sai\_object\_id\_t \*objects = calloc (set\_attr.value.objlist.object\_count, sizeof(sai\_object\_list\_t));

objects[0] = mirror\_span\_object;

objects[1] = mirror\_rspan\_object;

set\_attr.value.objlist.object\_list = objects;

sai\_port\_api-> set\_port\_attribute(port\_object\_id, &set\_attr);

get\_attr.id = SAI\_PORT\_ATTR\_EGRESS\_MIRROR\_SESSION;

get\_attr.value.objlist.object\_count = 1;

get\_attr.value.objlist.object\_list = calloc (get\_attr.value.objlist.object\_count, sizeof(sai\_object\_list\_t))

sai\_port\_api-> get\_port\_attribute(port\_object\_id, 1, &get\_attr);

//Check if SAI\_STATUS\_BUFFER\_OVERFLOW error is returned. If so invoke the get port attribute after reallocing the memory with the required object count in get\_attr.value.objlist.object\_count

get\_attr.value.objlist.object\_list = calloc (get\_attr.value.objlist.object\_count, sizeof(sai\_object\_list\_t))

sai\_port\_api-> get\_port\_attribute(port\_object\_id, 1, &get\_attr);

Check if get\_attr.value.objlist.object\_list[0] is mirror\_span\_object

Check if get\_attr.value.objlist.object\_list[1] is mirror\_rspan\_object

## Remove SPAN mirror session alone on the same port for egress mirroring (update of the object\_list scenario)

sai\_api\_query(SAI\_API\_PORT, &sai\_port\_api);

sai\_object\_id\_t port\_object\_id = 1 //Embed the port in the port numbering format.

sai\_attribute\_t set\_attr;

sai\_attribute\_t get\_attr;

set\_attr.id = SAI\_PORT\_ATTR\_EGRESS\_MIRROR\_SESSION;

set\_attr.value.objlist.object\_count = 1;

sai\_object\_id\_t \*objects = calloc (set\_attr.value.objlist.object\_count, sizeof(sai\_object\_list\_t));

objects[0] = mirror\_rspan\_object;

set\_attr.value.objlist.object\_list = objects;

sai\_port\_api-> set\_port\_attribute(port\_object\_id, &set\_attr);

get\_attr.id = SAI\_PORT\_ATTR\_EGRESS\_MIRROR\_SESSION;

get\_attr.value.objlist.object\_count = 1;

get\_attr.value.objlist.object\_list = calloc (get\_attr.value.objlist.object\_count, sizeof(sai\_object\_list\_t))

sai\_port\_api-> get\_port\_attribute(port\_object\_id, 1, &get\_attr);

Check if get\_attr.value.objlist.object\_list[0] is mirror\_rspan\_object

## Remove mirror session

sai\_mirror\_api->remove\_mirror\_session (mirror\_span\_object);

Check if SAI\_STATUS\_FAILURE is returned as mirror ports are still attached to the session.

Disassociate all the ports attached to the session and check if the session is removed successfully.

## Create ERSPAN session

sai\_api\_query(SAI\_API\_MIRROR, &sai\_mirror\_api);

sai\_object\_id\_t mirror\_erspan\_object;

sai\_attribute\_t attr[3] = {0};

attr[0].id = SAI\_MIRROR\_SESSION\_ATTR\_TC;

attr[0].value.u8 = 7;

attr[1].id = SAI\_MIRROR\_SESSION\_ATTR\_MONITOR\_PORT;

attr[1].value.oid = 9; //Embed in sai\_object\_id port numbering format

attr[2].id = SAI\_MIRROR\_SESSION\_ATTR\_TYPE;

attr[2].value.s32 = SAI\_MIRROR\_TYPE\_ENHANCED\_REMOTE;

attr[3].id =SAI\_MIRROR\_SESSION\_ATTR\_VLAN\_TPID;

attr[3].value.u16 = 0x8100;

attr[4].id =SAI\_MIRROR\_SESSION\_ATTR\_VLAN\_ID;

attr[4].value.u16 = 0x2;

attr[5].id =SAI\_MIRROR\_SESSION\_ATTR\_VLAN\_PRI;

attr[5].value.u8 = 0x6;

attr[6].id =SAI\_MIRROR\_SESSION\_ATTR\_ENCAP\_TYPE;

attr[6].value.s32 = 0x1;

attr[7].id =SAI\_MIRROR\_SESSION\_ATTR\_IPHDR\_VERSION;

attr[7].value.u8 = 0x4;

attr[8].id =SAI\_MIRROR\_SESSION\_ATTR\_TOS;

attr[8].value.u16 = 0x2;

attr[9].id =SAI\_MIRROR\_SESSION\_ATTR\_TTL;

attr[9].value.u8 = 0x2;

attr[10].id =SAI\_MIRROR\_SESSION\_ATTR\_SRC\_IP\_ADDRESS;

attr[10].value.ipaddr.addr.ip4 = 0xa0101002;

attr[10].value.ipaddr.addr\_family = SAI\_IP\_ADDR\_FAMILY\_IPV4;

attr[11].id =SAI\_MIRROR\_SESSION\_ATTR\_DST\_IP\_ADDRESS;

attr[11].value.ipaddr.addr.ip4 = 0xa0101011;

attr[11].value.ipaddr.addr\_family = SAI\_IP\_ADDR\_FAMILY\_IPV4;

attr[12].id =SAI\_MIRROR\_SESSION\_ATTR\_SRC\_MAC\_ADDRESS;

attr[12].value.mac[0] = 0x00;

attr[12].value.mac[1] = 0x01;

attr[12].value.mac[2] = 0x02;

attr[12].value.mac[3] = 0x03;

attr[12].value.mac[4] = 0x04;

attr[12].value.mac[5] = 0x05;

attr[13].id =SAI\_MIRROR\_SESSION\_ATTR\_DST\_MAC\_ADDRESS;

attr[13].value.mac[0] = 0x00;

attr[13].value.mac[1] = 0x11;

attr[13].value.mac[2] = 0x12;

attr[13].value.mac[3] = 0x13;

attr[13].value.mac[4] = 0x14;

attr[13].value.mac[5] = 0x15;

attr[14].id =SAI\_MIRROR\_SESSION\_ATTR\_GRE\_PROTOCOL\_TYPE;

attr[14].value.u16 = 0x88be;

sai\_mirror\_api->create\_mirror\_session (&mirror\_erspan\_object, 15,

attr);

## Associate the ERSPAN to a acl rule

sai\_api\_query(SAI\_API\_ACL; &sai\_acl\_api);

sai\_acl\_table\_id\_t acl\_table\_id;

sai\_acl\_entry\_id\_t acl\_rule\_id;

sai\_attribute\_t attr[12] = {0};

attr[0].id = SAI\_ACL\_TABLE\_ATTR\_STAGE;

attr[0].value.s32= 0;

attr[1].id = SAI\_ACL\_TABLE\_ATTR\_PRIORITY;

attr[1].value.u32 = 1;

attr[2].id = SAI\_ACL\_TABLE\_ATTR\_FIELD\_SRC\_MAC;

attr[3].id = SAI\_ACL\_TABLE\_ATTR\_FIELD\_DST\_MAC;

attr[4].id = SAI\_ACL\_TABLE\_ATTR\_FIELD\_ETHER\_TYPE;

attr[5].id = SAI\_ACL\_TABLE\_ATTR\_FIELD\_IP\_TYPE;

attr[6].id = SAI\_ACL\_TABLE\_ATTR\_FIELD\_INNER\_VLAN\_ID;

attr[7].id = SAI\_ACL\_TABLE\_ATTR\_FIELD\_INNER\_VLAN\_PRI;

attr[8].id = SAI\_ACL\_TABLE\_ATTR\_FIELD\_INNER\_VLAN\_CFI;

attr[9].id = SAI\_ACL\_TABLE\_ATTR\_FIELD\_IP\_PROTOCOL;

attr[10].id = SAI\_ACL\_ENTRY\_ATTR\_FIELD\_IN\_PORT;

attr[11].id = SAI\_ACL\_ENTRY\_ATTR\_FIELD\_IN\_PORTS;

sai\_acl\_api->create\_acl\_table (&acl\_table\_id, 12, &attr);

sai\_attribute\_t rule\_attr[10] = {0};

rule\_attr[0].id = SAI\_ACL\_ENTRY\_ATTR\_TABLE\_ID;

rule\_attr[0].value.u32= acl\_table\_id;

rule\_attr[1].id = SAI\_ACL\_ENTRY\_ATTR\_PRIORITY;

rule\_attr[1].value.u32 = 1;

rule\_attr[2].id = SAI\_ACL\_ENTRY\_ATTR\_ADMIN\_STATE;

rule\_attr[2].value.booldata= true;

rule\_attr[3].id = SAI\_ACL\_ENTRY\_ATTR\_FIELD\_IN\_PORTS;

rule\_attr[3].value.portlist.port\_count = 2;

rule\_attr[3].value.portlist.port\_list = (sai\_port\_id\_t \*) calloc(

2, sizeof(sai\_port\_id\_t));

rule\_attr[3].value.portlist.port\_list[0] = 1;//Embed in sai\_object\_id port numbering format

rule\_attr[3].value.portlist.port\_list[1] = 5;// Embed in sai\_object\_id port numbering format

rule\_attr[4].id = SAI\_ACL\_ENTRY\_ATTR\_ACTION\_MIRROR\_INGRESS;

rule\_attr[4].value.u32 = mirror\_erspan\_object;

sai\_acl\_api->create\_acl\_rule (&acl\_rule\_id, 5, &rule\_attr);

## Create ERSPAN session without a mandatory attribute

sai\_api\_query(SAI\_API\_MIRROR, &sai\_mirror\_api);

sai\_object\_id\_t mirror\_erspan\_object;

sai\_attribute\_t attr[15] = {0};

attr[0].id = SAI\_MIRROR\_SESSION\_ATTR\_TC;

attr[0].value.u8 = 7;

attr[1].id = SAI\_MIRROR\_SESSION\_ATTR\_MONITOR\_PORT;

attr[1].value.oid = 9; //Embed in sai\_object\_id port numbering format

attr[2].id = SAI\_MIRROR\_SESSION\_ATTR\_TYPE;

attr[2].value.s32 = SAI\_MIRROR\_TYPE\_ENHANCED\_REMOTE;

attr[3].id =SAI\_MIRROR\_SESSION\_ATTR\_VLAN\_TPID;

attr[3].value.u16 = 0x8100;

attr[4].id =SAI\_MIRROR\_SESSION\_ATTR\_VLAN\_ID;

attr[4].value.u16 = 0x2;

attr[5].id =SAI\_MIRROR\_SESSION\_ATTR\_VLAN\_PRI;

attr[5].value.u8 = 0x6;

attr[6].id =SAI\_MIRROR\_SESSION\_ATTR\_ENCAP\_TYPE;

attr[6].value.s32 = 0x1;

attr[7].id =SAI\_MIRROR\_SESSION\_ATTR\_IPHDR\_VERSION;

attr[7].value.u8 = 0x4;

attr[8].id =SAI\_MIRROR\_SESSION\_ATTR\_TOS;

attr[8].value.u16 = 0x2;

attr[9].id =SAI\_MIRROR\_SESSION\_ATTR\_TTL;

attr[9].value.u8 = 0x2;

attr[10].id =SAI\_MIRROR\_SESSION\_ATTR\_SRC\_IP\_ADDRESS;

attr[10].value.ipaddr.addr.ip4 = 0xa0101002;

attr[10].value.ipaddr.addr\_family = SAI\_IP\_ADDR\_FAMILY\_IPV4;

attr[11].id =SAI\_MIRROR\_SESSION\_ATTR\_DST\_IP\_ADDRESS;

attr[11].value.ipaddr.addr.ip4 = 0xa0101011;

attr[11].value.ipaddr.addr\_family = SAI\_IP\_ADDR\_FAMILY\_IPV4;

attr[12].id =SAI\_MIRROR\_SESSION\_ATTR\_SRC\_MAC\_ADDRESS;

attr[12].value.mac[0] = 0x00;

attr[12].value.mac[1] = 0x01;

attr[12].value.mac[2] = 0x02;

attr[12].value.mac[3] = 0x03;

attr[12].value.mac[4] = 0x04;

attr[12].value.mac[5] = 0x05;

attr[13].id =SAI\_MIRROR\_SESSION\_ATTR\_DST\_MAC\_ADDRESS;

attr[13].value.mac[0] = 0x00;

attr[13].value.mac[1] = 0x11;

attr[13].value.mac[2] = 0x12;

attr[13].value.mac[3] = 0x13;

attr[13].value.mac[4] = 0x14;

attr[13].value.mac[5] = 0x15;

sai\_mirror\_api->create\_mirror\_session (&mirror\_erspan\_object, 14,

attr);

Check if SAI\_MANDATORY\_ATTRIBUTE\_MISSING status is returned.

Repeat the same with all the other attributes. And all other span types.

# Appendix

Message: 1

Date: Thu, 19 Mar 2015 08:39:51 +0000

I am fine with 2, but just want to know what kind of L2 tunnel the NPU is going to use, hence the tunnel header used for encap the packet. So that the user can know what the fields are needed to be filled as attributes.

For example, the SAI\_MIRROR\_L3\_GRE\_TUNNEL is more clear to me, we know the encapsulation header is IP header plus the GRE header.

Subject: RE: [Opencompute-networking] SAI - Mirroring and SFLOW API's for review

Hi Guohan,

1 - Yes they need not be READ ONLY. It can be R/W. We will modify

2 - Some NPU's support only L2 based tunnel header for ERSPAN and hence has added that

-Arun

**Subject:** Re: [Opencompute-networking] SAI - Mirroring and SFLOW API's for review

All the mirror session attributes are CREATE + READ-ONLY now. Maybe except for SAI\_MIRROR\_SESSION\_ATTR\_TYPE and SAI\_MIRROR\_SESSION\_ATTR\_ENCAP\_TYPE, all others should allow to be modified after the session is created?

For SAI\_MIRROR\_L2\_TUNNEL, which L2 tunnel is going to use to encap the packet? What’s the packet format for the mirror packet?

-Guohan

**Subject:** Re: [Opencompute-networking] SAI - Mirroring and SFLOW API's for review

Updated spec

<https://github.com/opencomputeproject/OCP-Networking-Project-Community-Contributions/blob/master/sai/doc/Mirror/SAI-Proposal-2-Mirror-Ver3.docx>

We would use this to generate headers

-Arun

**Subject:** Re: [Opencompute-networking] SAI - Mirroring and SFLOW API's for review

Requesting some minor clarifications before the headers are released..

The document talks about the mandatory functions like:

“Adding source ports to the session”.

“Removing the source ports from the session”.

But then in the attribute list there is no mention of “source ports”.

If the naming convention could be matched to help clarify the usage, that will be great.

-thanks

Tushar

**Subject:** RE: [Opencompute-networking] SAI - Mirroring and SFLOW API's for review

Updated spec

<https://github.com/opencomputeproject/OCP-Networking-Project-Community-Contributions/blob/master/sai/doc/Mirror/SAI-Proposal-2-Mirror-Ver3.docx>

We would use this to generate headers

-Arun

Subject: RE: [Opencompute-networking] SAI - Mirroring and SFLOW API's for review

Yes agreed will make the change

-Arun

From: Tushar Tyagi [<mailto:tusharty@broadcom.com>]   
Sent: Tuesday, March 17, 2015 2:48 AM  
To: Manickam, Arunsubash; [opencompute-networking@lists.opencompute.org](mailto:opencompute-networking@lists.opencompute.org)  
Subject: RE: [Opencompute-networking] SAI - Mirroring and SFLOW API's for review

Hello Arun,

A small input..

Looking at other sai object create functions, I think that for maintaining consistency in..

*typedef sai\_status\_t (\*sai\_create\_mirror\_session\_fn)(*

*\_Inout\_ sai\_mirror\_session\_id\_t \*session\_id,*

*\_In\_ sai\_mirror\_type\_t mirror\_type,*

*\_In\_ uint32\_t attr\_count,*

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

**sai\_mirror\_type\_t mirror\_type** should be moved into the **attr\_list**.

-thanks

Tushar