

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

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

**Contents**

[List of Changes i](#_Toc410923319)

[1 Overview 1](#_Toc410923320)

[2 Specification 1](#_Toc410923321)

[2.1 Changes to sai.h 1](#_Toc410923322)

[2.2 Changes to saitypes.h 1](#_Toc410923323)

[2.3 New definitions in saisflow.h 1](#_Toc410923324)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 0.9.2 | Proposal for sFlow |  | 2/5/15 |

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

Proposal for saisflow.h interface

# Specification

**Mandatory**

Implements sFlow management functions like

* Creating a sFlow session
* Destroying the sFlow session
* Adding ports to the session
* Removing ports from the session

## Changes to sai.h

typedef enum \_sai\_api\_t {

SAI\_API\_SFLOW= 15, /\* sai\_sflow\_api\_t \*

} sai\_api\_t;

## Changes to saitypes.h

#define UINT32 sai\_sflow\_session\_id\_t

## New definitions in saisflow.h

typedef enum \_sai\_sflow\_packet\_type\_t

{

SAI\_SFLOW\_PACKET\_TYPE\_UNICAST = (1 << 0),

SAI\_SFLOW\_PACKET\_TYPE\_MULTICAST= (1 << 1),

SAI\_SFLOW\_PACKET\_TYPE\_BROADCAST = (1 << 2),

SAI\_SFLOW\_PACKET\_TYPE\_UNKNOWN\_UNICAST = (1 << 3),

SAI\_SFLOW\_PACKET\_TYPE\_UNREGISTERED\_MULTICAST = (1 << 4),

SAI\_SFLOW\_PACKET\_TYPE\_ALL = (1 << 5),

} **sai\_sflow\_packet\_type\_t**;

typedef enum \_sai\_sflow\_direction\_t

{

/\* Ingress sFlow \*/

SAI\_SFLOW\_DIR\_INGRESS,

/\* Egress sFlow \*/

SAI\_SFLOW\_DIR\_EGRESS,

/\*Both Ingress and Egress sFlow\*/

SAI\_SFLOW\_DIR\_INGRESS\_EGRESS,

} **sai\_sflow\_direction\_t**;

typedef enum \_sai\_sflow\_type\_t

{

/\* Copy the sample packets to CPU \*/

SAI\_SFLOW\_SLOW\_PATH,

/\* Mirror the sample packets by MTP\_index\*/

SAI\_SFLOW\_FAST\_PATH,

} **sai\_sflow\_type\_t**;

typedef enum \_sai\_sflow\_attr\_t

{

/\* CREATE + READ-WRITE \*/

/\* sFlow direction of type sai\_sflow\_direction\_t \*/

SAI\_SFLOW\_ATTR\_DIRECTION,

/\* CREATE + READ-WRITE \*/

/\* sFlow sampling rate of type uint32 \*/

SAI\_SFLOW\_ATTR\_SAMPLE\_RATE,

/\* CREATE + READ-WRITE \*/

/\* sFlow Enable/Disable of type bool \*/

SAI\_SFLOW\_ATTR\_ADMIN\_STATE,

/\* CREATE + READ-WRITE \*/

/\* Packet Types to be sampled of type sai\_sflow\_packet\_type\_t. Unicast, multicast, broadcast

Unknown unicast or unknown multicast. Either one or all can be chosen\*/

SAI\_SFLOW\_ATTR\_PACKET\_TYPE,

/\* CREATE + READ-ONLY \*/

/\* sFlow switching type sai\_sflow\_type. Default would be SAI\_SFLOW\_SLOW\_PATH \*/

SAI\_SFLOW\_ATTR\_TYPE,

/\* All the attributes below are mandatory for SAI\_SFLOW\_FAST\_PATH \*/

/\* CREATE + READ-ONLY \*/

/\* sFlow Source MAC Address [sai\_mac\_t]\*/

SAI\_SFLOW\_ATTR\_SRC\_MAC\_ADDRESS,

/\* CREATE + READ-ONLY \*/

/\* sFlow Destination MAC Address [sai\_mac\_t]\*/

SAI\_SFLOW\_ATTR\_DST\_MAC\_ADDRESS,

/\* CREATE + READ-ONLY \*/

/\* sFlow VLAN Id [sai\_vlan\_id\_t]\*/

SAI\_SFLOW\_ATTR\_VLAN\_ID,

/\* CREATE + READ-ONLY \*/

/\* sFlow collector’s next hop information. Destination port shall be retrieved by that information [sai\_port\_id\_t]\*/

SAI\_SFLOW\_ATTR\_DST\_PORT,

/\* CREATE + READ-ONLY \*/

/\* sFlow collector address required for hardware switching [sai\_ip\_address\_t] \*/

SAI\_SFLOW\_ATTR\_SRC\_IP\_ADDRESS,

/\* CREATE + READ-ONLY \*/

/\* sFlow agent address required for hardware switching [sai\_ip\_address\_t ] \*/

SAI\_SFLOW\_ATTR\_DST\_IP\_ADDRESS,

/\* CREATE + READ-ONLY \*/

/\* sFlow Header UDP information [uint16\_t] \*/

SAI\_SFLOW\_ATTR\_L4\_UDP\_DST\_PORT,

/\* CREATE + READ-ONLY \*/

/\* sFlow Header UDP source port [uint16\_t] \*/

SAI\_SFLOW\_ATTR\_L4\_UDP\_SRC\_PORT,

/\* CREATE + READ-ONLY \*/

/\* sFlow Header Datagram size [uint16\_t] \*/

SAI\_SFLOW\_ATTR\_L4\_UDP\_DATAGRAM\_SIZE,

} **sai\_sflow\_attr\_t**;

/\*

\* Routine Description:

\* Create sFlow session.

\*

\* Arguments:

\* [in] session\_id - sflow session\_id

\* [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\_sflow\_session\_fn)(

\_Inout\_ sai\_sflow\_session\_id\_t session\_id,

\_In\_ uint32\_t attr\_count,

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

/\*

\* Routine Description:

\* Delete sFlow session.

\*

\* Arguments:

\* [in] session\_id - sflow session\_id

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_delete\_sflow\_session\_fn)(

\_In\_ sai\_sflow\_session\_id\_t session\_id);

/\*

\* Routine Description:

\* Add sFlow port to a a session

\*

\* Arguments:

\* [in] session\_id - sFlow session

\* [in] port\_id - port on which sFlow is to be enabled

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_add\_ports\_to\_sflow\_session\_fn)(

\_In\_ sai\_sflow\_session\_id\_t session\_id,

\_In\_ sai\_port\_id\_t port\_id);

/\*

\* Routine Description:

\* Remove sFlow port from a session

\*

\* Arguments:

\* [in] session\_id - sFlow session

\* [in] port\_id - port to be removed from sFlow session

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_ports\_from\_session\_fn)(

\_In\_ sai\_sflow\_session\_id\_t session\_id,

\_In\_ sai\_port\_id\_t port\_id);

/\*

\* Routine Description:

\* Set sFlow attributes on a port.

\*

\* Arguments:

\* [in] session\_id – session on which sflow should be set

\* [in] attr - attribute value

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

\_In\_ sai\_sflow\_session\_id\_t session\_id,

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

/\*

\* Routine Description:

\* Get sFlow attributes on a port.

\*

\* Arguments:

\* [in] session\_id - session on which sflow attributes should bre

\* [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\_sflow\_attribute\_fn)(

\_In\_ sai\_sflow\_session\_id\_t session\_id,

\_In\_ uint32\_t attr\_count,

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

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

typedef struct \_sai\_sflow\_api\_t

{

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

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

sai\_add\_ports\_to\_sflow\_session\_fn add\_sflow\_port;

sai\_remove\_ports\_from\_sflow\_session\_fn remove\_sflow\_port;

sai\_set\_sflow\_attribute\_fn set\_sflow\_attribute;

sai\_get\_sflow\_attribute\_fn get\_sflow\_attribute;

} **sai\_sflow\_api\_t**;