

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **LAG and HASH API’s** |
| **Authors** | **DELL** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **1/23/2015** |
| **SAI-Version** | **V0.9.2** |

**Contents**

[List of Changes i](#_Toc411243390)

[1 Overview 1](#_Toc411243391)

[2 Specification 1](#_Toc411243392)

[2.1 LAG 1](#_Toc411243393)

[2.2 HASH 4](#_Toc411243394)

[2.2.1 New Definitions 4](#_Toc411243395)

[2.2.2 New attributes added to existing SWITCH attributes 4](#_Toc411243396)

[2.2.3 Definitions to be removed from existing header file 5](#_Toc411243397)

[3 Appendix 5](#_Toc411243398)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 0.9.2 | Proposal for LAG, HASH – Version 1 |  | 1/23/15 |
| 0.9.2 | Version 2   * Incorporated feedback from Mellanox. Appendix section has details of the changes * Added details on ordering a port addition to a LAG |  | 02/16/15 |

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

The document covers details of LAG object – Creation, deletion, update. Currently, LAG is defined as a container for the SAI ports. There is no attributes defined for the LAG object. We would define the get/set attribute API’s for future use.

The LAG ID returned is from the SAI Port range space. We may want to support the LAG ID being used for get/set select port attributes in future.

The ports added to a LAG can be added to the port list in ascending order of the port number. This helps to make sure the hash is retained when we add/remove a port from a LAG due to port link status change

The document also defines hash attributes that can be used to configure LAG.

# Specification

## LAG

typedef enum \_sai\_api\_t {

SAI\_API\_LAG= 12, /\* sai\_lag\_api\_t \*

}

/\*

\* Routine Description:

\* Create LAG interface.

\* Port type (sai\_port\_type\_t) of LAG interface will be  SAI\_PORT\_TYPE\_LAG

\* Arguments:

\* [out] lag\_id - LAG 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\_lag\_interface\_fn)(**

**\_Out\_ sai\_port\_id\_t\* lag\_id,**

**\_In\_ int attr\_count,**

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

**);**

/\*

\* Routine Description:

\* Remove LAG interface

\*

\* Arguments:

\* [in] lag\_id - lag id

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

**typedef sai\_status\_t(\*sai\_remove\_lag\_interface\_fn)(**

**\_In\_ sai\_port\_id\_t lag\_id**

**);**

/\*

\* Routine Description:

\* Add Port to LAG

\*

\* Arguments:

\* [in] lag\_id - LAG id

\* [in] port\_count - number of ports

\* [in] port\_list - pointer to membership structures

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

**\_In\_ sai\_port\_id\_t lag\_id,**

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

**\_In\_ const sai\_port\_id\_t\* port\_list**

**);**

/\*

\* Routine Description:

\* Remove Port from LAG

\*

\* Arguments:

\* [in] lag\_id - LAG id

\* [in] port\_count - number of ports

\* [in] port\_list - pointer to membership structures

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

**\_In\_ sai\_port\_id\_t lag\_id,**

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

**\_In\_ const sai\_port\_id\_t\* port\_list**

**);**

/\*

\* Routine Description:

\* Set LAG interface attribute

\*

\* Arguments:

\* [in]lag\_id– lag id

\* [in] attr - attribute

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

**typedef sai\_status\_t (\*sai\_set\_lag\_interface\_attribute\_fn)(**

**\_In\_ sai\_port\_id\_t lag\_id,**

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

**);**

/\*

\* Routine Description:

\* Get LAG interface attribute

\*

\* Arguments:

\* [in] lag\_id- lag\_id

\* [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\_lag\_interface\_attribute\_fn)(**

**\_In\_ sai\_port\_id\_t lag\_id,**

**\_In\_ int attr\_count,**

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

**);**

/\*

\* LAG methods table retrieved with sai\_api\_query()

\*/

**typedef struct \_sai\_lag\_api\_t**

**{**

**sai\_create\_lag\_interface\_fn   create\_lag\_interface;**

**sai\_remove\_lag\_interface\_fn remove\_lag\_interface;**

**sai\_set\_lag\_attribute\_fn set\_lag\_attribute;**

**sai\_get\_lag\_attribute\_fn get\_lag\_attribute;**

**sai\_add\_ports\_to\_lag\_fn add\_ports\_to\_lag;**

**sai\_remove\_ports\_from\_lag\_fn remove\_ports\_from\_lag;**

**} sai\_lag\_api\_t;**

## HASH

ECMP and LAG shall use the system wide hash logic for flow distribution over the links. Hash configuration is defined per switch for each application like ECMP and LAG. Hash calculation will use following attributes for flow distribution over LAG and ECMP

* Hash Seed
* Hash algorithm
* Packet hash fields

Since Hash by itself is not an object and does not have a handle, we cannot have set/get API’s for hash using the Hash Type(LAG, ECMP) as an handle. Hence we would add these to the switch attributes duplicating entries for LAG and ECMP

### New Definitions

**Changes to saitypes.h**

typedef uint32\_t  sai\_switch\_hash\_seed\_t; /\* hash seed value\*/

/\*

\* List of hash fields to set

\*/

typedef struct \_sai\_switch\_hash\_fields\_t

{

uint32\_t num\_fields ;

sai\_switch\_hash\_field\_types\_t \*field\_list ;

} sai\_switch\_hash\_fields\_t ;

**Changes to saiswitch.h**

typedef enum \_sai\_switch\_hash\_field\_types\_t  
{  
    SAI\_HASH\_SRC\_IP,      
    SAI\_HASH\_DST\_IP,          
    SAI\_HASH\_VLAN\_ID,

    SAI\_HASH\_IP\_PROTOCOL,        
    SAI\_HASH\_ETHERTYPE,       
    SAI\_HASH\_L4\_SOURCE\_PORT,   
    SAI\_HASH\_L4\_DEST\_PORT,  
    SAI\_HASH\_SOURCE\_MAC ,      
    SAI\_HASH\_DEST\_MAC ,         
 SAI\_HASH\_DEST\_PORT ,      
    SAI\_HASH\_SOURCE\_PORT        
     
} sai\_switch\_hash\_field\_types\_t;

typedef enum \_sai\_switch\_hash\_algo\_t {

SAI\_HASH\_XOR ,

SAI\_HASH\_CRC,

SAI\_HASH\_RANDOM,

SAI\_HASH\_SYMMETRIC            
} sai\_switch\_hash\_algo\_t;

### New attributes added to existing SWITCH attributes

typedef enum \_sai\_switch\_hash\_attr\_t  
{  
    SAI\_SWITCH\_ ATTR\_LAG\_HASH\_ALGO ,   /\*value sai\_switch\_hash\_algo\_t\*/

SAI\_SWITCH\_ATTR\_LAG\_HASH\_ SEED ,   /\*value sai\_switch\_hash\_seed\_t \*/

SAI\_SWITCH\_ ATTR\_LAG\_HASH\_FIELDS,     /\*value sai\_switch\_hash\_fields\_t ,

sai\_switch\_hash\_field\_types\_t \*/

SAI\_SWITCH\_ ATTR\_ECMP\_HASH\_ALGO ,   /\*value sai\_switch\_hash\_algo\_t\*/

SAI\_SWITCH\_ATTR\_ECMP\_HASH\_ SEED ,   /\*value sai\_switch\_hash\_seed\_t \*/

SAI\_SWITCH\_ ATTR\_ECMP\_HASH\_FIELDS,     /\*value sai\_switch\_hash\_fields\_t,

sai\_switch\_hash\_field\_types\_t \*/

}

### Definitions to be removed from existing header file

ECMP hashing type [sai\_switch\_ecmp\_hash\_type\_t] \*/

SAI\_SWITCH\_ATTR\_ECMP\_HASH\_TYPE,

/\* ECMP hashing fields [sai\_switch\_ecmp\_hash\_fields\_t] \*/

SAI\_SWITCH\_ATTR\_ECMP\_HASH\_FIELDS,

/\*

\* Attribute data for SAI\_SWITCH\_ECMP\_HASH\_TYPE

\*/

typedef enum \_sai\_switch\_ecmp\_hash\_type\_t

{

SAI\_SWITCH\_ECMP\_HASH\_TYPE\_XOR,

SAI\_SWITCH\_ECMP\_HASH\_TYPE\_CRC,

} sai\_switch\_ecmp\_hash\_type\_t;

typedef enum \_sai\_switch\_ecmp\_hash\_fields\_t

{

SAI\_SWITCH\_ECMP\_HASH\_SRC\_IP = (1 << 0),

SAI\_SWITCH\_ECMP\_HASH\_DST\_IP = (1 << 1),

SAI\_SWITCH\_ECMP\_HASH\_L4\_SRC\_PORT = (1 << 2),

SAI\_SWITCH\_ECMP\_HASH\_L4\_DST\_PORT = (1 << 3),

} sai\_switch\_ecmp\_hash\_fields\_t;

# Appendix

**Review Comments – Matty, Mellanox – 01/30/2015**

**From:** opencompute-networking-bounces@lists.opencompute.org [mailto:opencompute-networking-bounces@lists.opencompute.org] **On Behalf Of** Manickam, Arunsubash  
**Sent:** Saturday, January 31, 2015 5:43 AM  
**To:** mattyk@mellanox.com; opencompute-networking@lists.opencompute.org  
**Subject:** Re: [Opencompute-networking] SAI - LAG, HASH API Proposal

**Dell - Internal Use - Confidential**

Thanks for your feedback Matty

1 –

Remove\_all API -

I this out API should look the same since we don’t have delete\_all in router, net hop, neigh …

I suggest we remove it from LAG has well

Agreed I will remove it in my next update

2 –

All those HASH algo look like vendor specific implementation and not generic

I think we should it should be

SAI\_HASH\_CRC

SAI\_HASH\_XOR

SAI\_HASH\_RANDOM

In addition should have another HASH attribute SAI\_HASH\_ SYMMETRIC

In order to be able to use symmetric hash

Yes I agree we can reduce the hash list. Do you think the 3 above + Symmetric should be bare minimum we expect from a NPU. How about load based hashing

3 –

Some NPU support LAG hash per LAG

And ECMP hash per VRF /route

IMO we should have those attribute on LAG,VRF as well .

Yes we could add that in future. In that case the same set of attributes get duplicated or we define a attribute that is common. One suggestion would help in this case is to define HASH in a separate header file and have functions where HASH can be set for different objects – Switch, LAG, VRF. That would look clean and avoid duplicating the definitions

4 –

I think that we should have a single attribute for  hash fields

SAI\_SWITCH\_ ATTR\_LAG\_HASH\_FIELDS

SAI\_SWITCH\_ ATTR\_ECMP\_HASH\_FIELDS

In order to configure hash  on MAC,ipV4, MPLS,,Vxlan all you need to do it to call the set attribute with list on attribute

Agreed I will make the change

5 –

IMO we should not use  bit field

The way SAI should handle multiple value on thesame attribute

is by providing a list of  e attribute.(to same approch we took in ACL when use rule can match on multiple fields)

Yes we could follow the same approach as ACL to be consistent

Once we converge on #2 and #3 I will make the updates and send a revised proposal

Thanks,

Arun