

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **L2MC** |
| **Authors** | **CENTEC** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **17/10/2015** |
| **SAI-Version** | **0.9.3** |

**Contents**

[List of Changes i](#_Toc459910430)

[1 Overview 1](#_Toc459910431)

[2 Specification 1](#_Toc459910432)

[2.1 Change to sai.h 1](#_Toc459910433)

[2.2 Change to saitypes.h 1](#_Toc459910434)

[2.3 Change to sainexthopgroup.h 1](#_Toc459910435)

[2.4 Changes to saifdb.h 6](#_Toc459910436)

[2.5 Changes to saivlan.h 7](#_Toc459910437)

[3 Examples 8](#_Toc459910438)

[3.1 Create a generic group 8](#_Toc459910439)

[3.2 Create group members 8](#_Toc459910440)

[3.3 Create a multicast fdb entry associated with the group 8](#_Toc459910441)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| Initial Version | Base version |  | 17/10/2015 |
| 0.2 | Changes after meeting review |  | 01/07/2016 |
| 0.7 | Extend FDB to implement L2Multicast | Min Yao | 20/07/2016 |
| 0.8 | * change the next\_hop\_group to some group containers for general purpose that the container contains a group of objects and has the attribute type e.g. ECMP or L2MC. * Make a generic group type that defines the list of port for multicast FDB entry * Add Switch-level attribute to define whether switch supports IP based L2 multicast. (RO) | Min Yao | 25/08/2016 |

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

L2 Multicast, a basic layer 2 feature, which provides a security solution by isolating flooding domain from VLAN to l2multicast group. Meanwhile, it decreases the traffic load on a Layer 2 switch. L2 multicast can be implemented by multicast fdb lookup table (referred to as MAC based L2 multicast) or SG/\*G lookup table (referred to as IP based L2 multicast, known as snooping). This document covers the implementation of MAC based L2 multicast.

# Specification

This section describes the details of the proposed interface/API

## Change to sai.h

/\*\*

\*

\* Defined API sets have assigned ID's. If specific api method table changes

\* in any way (method signature, number of methods), a new ID needs to be

\* created (e.g. VLAN2) and old API still may need to be supported for

\* compatibility with older adapter hosts.

\*

\*/

typedef enum \_sai\_api\_t

{

…

~~SAI\_API\_NEXT\_HOP\_GROUP = 8, /\*\*< sai\_next\_hop\_group\_api\_t \*/~~

**SAI\_API\_GROUP = 8, /\*\*< sai\_group\_api\_t \*/**

…

} sai\_api\_t;

## Change to saitypes.h

/\*\*

\* @brief sai object type

\*/

typedef enum \_sai\_object\_type\_t {

…

~~SAI\_OBJECT\_TYPE\_NEXT\_HOP\_GROUP = 5,~~

**SAI\_OBJECT\_TYPE\_GROUP = 5,**

…

~~SAI\_OBJECT\_TYPE\_NEXT\_HOP\_GROUP\_MEMBER = 43,~~

**SAI\_OBJECT\_TYPE\_GROUP\_MEMBER = 43,**

…

} sai\_object\_type\_t;} sai\_api\_t;

## Change to sainexthopgroup.h

Rename file name to saigroup.h

/\*

\* Copyright (c) 2014 Microsoft Open Technologies, Inc.

\*

\* Licensed under the Apache License, Version 2.0 (the "License"); you may

\* not use this file except in compliance with the License. You may obtain

\* a copy of the License at http://www.apache.org/licenses/LICENSE-2.0

\*

\* THIS CODE IS PROVIDED ON AN \*AS IS\* BASIS, WITHOUT WARRANTIES OR

\* CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT

\* LIMITATION ANY IMPLIED WARRANTIES OR CONDITIONS OF TITLE, FITNESS

\* FOR A PARTICULAR PURPOSE, MERCHANTABLITY OR NON-INFRINGEMENT.

\*

\* See the Apache Version 2.0 License for specific language governing

\* permissions and limitations under the License.

\*

\* Microsoft would like to thank the following companies for their review and

\* assistance with these files: Intel Corporation, Mellanox Technologies Ltd,

\* Dell Products, L.P., Facebook, Inc

\*

\* Module Name:

\*

\* saigroup.h

\*

\* Abstract:

\*

\* This module defines generic SAI object Group API

\*

\*/

#if !defined (\_\_SAIGROUP\_H\_)

#define \_\_SAIGROUP\_H\_

#include <saitypes.h>

/\*\* \defgroup SAIGROUP SAI - Generic object group specific API definitions.

\*

\* \{

\*/

/\*\*

\* @brief Generic object group type

\*/

typedef enum \_sai\_group\_type\_t

{

/\*\* Group members are ECMP nexthops \*/

SAI\_GROUP\_TYPE\_ECMP,

**/\*\* Group members are outputs of a L2 multicast group \*/**

**SAI\_GROUP\_TYPE\_L2MC,**

/\*\* Other types of group to be defined in the future, e.g., WCMP \*/

} sai\_group\_type\_t;

/\*\*

\* @brief Attribute id for generic group

\*/

typedef enum \_sai\_group\_attr\_t

{

SAI\_GROUP\_ATTR\_START,

/\*\* READ-ONLY \*/

/\*\* Number of members in the group [uint32\_t] \*/

SAI\_GROUP\_ATTR\_MEMBER\_COUNT = SAI\_GROUP\_ATTR\_START,

/\*\* Member list [sai\_object\_list\_t] \*/

SAI\_GROUP\_ATTR\_MEMBER\_LIST,

/\*\* READ-WRITE \*/

/\*\* Generic group type [sai\_group\_type\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY) \*/

SAI\_GROUP\_ATTR\_TYPE,

SAI\_GROUP\_ATTR\_END,

/\* Custom range base value \*/

SAI\_GROUP\_ATTR\_CUSTOM\_RANGE\_BASE = 0x10000000

} sai\_group\_attr\_t;

typedef enum \_sai\_group\_member\_attr\_t

{

SAI\_GROUP\_MEMBER\_ATTR\_START,

/\*\* READ\_WRITE \*/

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

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

/\*\* Group Member Object ID [sai\_object\_id\_t] (MANDATORY\_ON\_CREATE|CREATE\_ONLY)

\* When group type is ECMP, the member should be a nexthop object

**\* When group type is L2MC, the member should be a port/LAG object** \*/

SAI\_GROUP\_MEMBER\_ATTR\_MEMBER\_ID,

/\*\* Member Weights [sai\_uint32\_t] (CREATE\_AND\_SET)

\* Applicable when group type is ECMP (default to 1) \*/

SAI\_GROUP\_MEMBER\_ATTR\_WEIGHT,

SAI\_GROUP\_MEMBER\_ATTR\_END,

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

SAI\_GROUP\_MEMBER\_ATTR\_CUSTOM\_RANGE\_BASE = 0x10000000

} sai\_group\_member\_attr\_t;

/\*\*

\* Routine Description:

\* @brief Create generic group

\*

\* Arguments:

\* @param[out] group\_id - generic group id

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

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

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

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

\_In\_ uint32\_t attr\_count,

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

);

/\*\*

\* Routine Description:

\* @brief Remove generic group

\*

\* Arguments:

\* @param[in] group\_id - generic group id

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

\_In\_ sai\_object\_id\_t group\_id

);

/\*\*

\* Routine Description:

\* @brief Set Generic Group attribute

\*

\* Arguments:

\* @param[in] sai\_object\_id\_t - generic group id

\* @param[in] attr - attribute

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

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

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

);

/\*\*

\* Routine Description:

\* @brief Get Generic Group attribute

\*

\* Arguments:

\* @param[in] sai\_object\_id\_t - group\_id

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

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

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

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

\_In\_ uint32\_t attr\_count,

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

);

/\*\*

\* Routine Description:

\* @brief Create generic group member

\*

\* Arguments:

\* @param[out] group\_member\_id - generic group member id

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

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

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_create\_group\_member\_fn)(

\_Out\_ sai\_object\_id\_t\* group\_member\_id,

\_In\_ uint32\_t attr\_count,

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

);

/\*\*

\* Routine Description:

\* @brief Remove generic group member

\*

\* Arguments:

\* @param[in] group\_member\_id - generic group member id

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_group\_member\_fn)(

\_In\_ sai\_object\_id\_t group\_member\_id

);

/\*\*

\* Routine Description:

\* @brief Set Generic Group Member attribute

\*

\* Arguments:

\* @param[in] sai\_object\_id\_t - group\_member\_id

\* @param[in] attr - attribute

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_group\_member\_attribute\_fn)(

\_In\_ sai\_object\_id\_t group\_member\_id,

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

);

/\*\*

\* Routine Description:

\* @brief Get Generic Group Member attribute

\*

\* Arguments:

\* @param[in] sai\_object\_id\_t - group\_member\_id

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

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

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_get\_group\_member\_attribute\_fn)(

\_In\_ sai\_object\_id\_t group\_member\_id,

\_In\_ uint32\_t attr\_count,

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

);

/\*\*

\* @brief Generic group methods table retrieved with sai\_api\_query()

\*/

typedef struct \_sai\_group\_api\_t

{

sai\_create\_group\_fn create\_group;

sai\_remove\_group\_fn remove\_group;

sai\_set\_group\_attribute\_fn set\_group\_attribute;

sai\_get\_group\_attribute\_fn get\_group\_attribute;

sai\_create\_group\_member\_fn create\_group\_member;

sai\_remove\_group\_member\_fn remove\_group\_member;

sai\_set\_group\_member\_attribute\_fn set\_next\_member\_attribute;

sai\_get\_group\_member\_attribute\_fn get\_next\_member\_attribute;

} sai\_group\_api\_t;

/\*\*

\* \}

\*/

#endif // \_\_SAIGROUP\_H\_

## Changes to saifdb.h

/\*\*

\* @brief Attribute Id for fdb entry

\*/

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

{

SAI\_FDB\_ENTRY\_ATTR\_START,

/\*\* READ-ONLY \*/

/\*\* READ-WRITE \*/

/\*\* FDB entry type [sai\_fdb\_entry\_type\_t] (MANDATORY\_ON\_CREATE|CREATE\_AND\_SET) \*/

SAI\_FDB\_ENTRY\_ATTR\_TYPE = SAI\_FDB\_ENTRY\_ATTR\_START,

~~/\*\* FDB entry port id [sai\_object\_id\_t] (MANDATORY\_ON\_CREATE|CREATE\_AND\_SET)~~

~~\* The port id here can refer to a generic port object such as SAI port object id,~~

~~\* SAI LAG object id and etc. or to a tunnel next hop object in case the entry is~~

~~\* l2 tunnel \*/~~

~~SAI\_FDB\_ENTRY\_ATTR\_PORT\_ID,~~

**/\*\* FDB entry output id [sai\_object\_id\_t]**

**\* The output id here can refer to a generic port object such as SAI port object id,**

**\* SAI LAG object id and etc. or to a tunnel next hop object in case the entry is**

**\* l2 tunnel.**

**\* It can also refer to a genric group with type SAI\_GROUP\_TYPE\_L2MC**

**\* (MANDATORY\_ON\_CREATE when SAI\_FDB\_ENTRY\_ATTR\_PACKET\_ACTION = SAI\_PACKET\_ACTION\_FORWARD)**

**\* (CREATE\_AND\_SET) \*/**

**SAI\_FDB\_ENTRY\_ATTR\_OUTPUT\_ID,**

/\*\* FDB entry packet action [sai\_packet\_action\_t] (MANDATORY\_ON\_CREATE|CREATE\_AND\_SET) \*/

SAI\_FDB\_ENTRY\_ATTR\_PACKET\_ACTION,

/\*\* User based Meta Data

\* [sai\_uint32\_t] (CREATE\_AND\_SET)

\* Value Range SAI\_SWITCH\_ATTR\_FDB\_DST\_USER\_META\_DATA\_RANGE \*/

SAI\_FDB\_ENTRY\_ATTR\_META\_DATA,

/\* -- \*/

SAI\_FDB\_ENTRY\_ATTR\_END,

/\* Custom range base value \*/

SAI\_FDB\_ENTRY\_ATTR\_CUSTOM\_RANGE\_START = 0x10000000,

/\* --\*/

SAI\_FDB\_ENTRY\_ATTR\_CUSTOM\_RANGE\_END

} sai\_fdb\_entry\_attr\_t;

## Changes to saivlan.h

/\*\*

\* @brief Attribute Id in sai\_set\_vlan\_attribute() and

\* sai\_get\_vlan\_attribute() calls

\*/

typedef enum \_sai\_vlan\_attr\_t

{

SAI\_VLAN\_ATTR\_START,

/\*\* READ-ONLY \*/

/\*\* List of vlan members in a VLAN [sai\_object\_list\_t]\*/

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

/\*\* READ-WRITE \*/

/\*\* Maximum number of learned MAC addresses [uint32\_t]

\* zero means learning limit disable. (default to zero).

\*/

SAI\_VLAN\_ATTR\_MAX\_LEARNED\_ADDRESSES,

/\*\* STP Instance that the VLAN is associated to [sai\_object\_id\_t]

\* (default to default stp instance id)\*/

SAI\_VLAN\_ATTR\_STP\_INSTANCE,

/\*\* To disable learning on a VLAN. [bool] (CREATE\_AND\_SET)

\* (default set to false)

\* This should override port learn settings. If this is set to true on a vlan,

\* then the source mac learning is disabled for this vlan on a member port even

\* if learn is enable on the port(based on port learn attribute)

\*/

SAI\_VLAN\_ATTR\_LEARN\_DISABLE,

**/\*\* VLAN Exception IGMP packet enable [bool]**

**\*(default is enabled)\*/**

**SAI\_VLAN\_ATTR\_EXCEPTION\_IGMP\_EN,**

**/\*\* VLAN Exception MLD packet enable [bool]**

**\*(default is enabled)\*/**

**SAI\_VLAN\_ATTR\_EXCEPTION\_MLD\_EN,**

**/\*\* group id, which contains the out interface list of unknown multicast**

**\* entry lookup**

**\* (CREATE\_AND\_SET)/**

**SAI\_VLAN\_ATTR\_UNKNOWN\_MCAST\_OUTPUT\_GROUP\_ID,**

/\*\* User based Meta Data

\* [sai\_uint32\_t] (CREATE\_AND\_SET)

\* Value Range SAI\_SWITCH\_ATTR\_VLAN\_USER\_META\_DATA\_RANGE \*/

SAI\_VLAN\_ATTR\_META\_DATA,

SAI\_VLAN\_ATTR\_END,

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

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

/\* --\*/

SAI\_VLAN\_ATTR\_CUSTOM\_RANGE\_END

} sai\_vlan\_attr\_t;

# Examples

Create a l2mc group with multiple output port and associate the l2mc group with multicast fdb entry.

## Create a generic group

sai\_api\_query(SAI\_API\_GROUP, &group\_api);

sai\_object\_id\_t l2mc\_group\_id = 0;

sai\_attribute\_t attr[2] = {0};

attr[0].id = SAI\_GROUP\_ATTR\_TYPE;

attr[0].value.s32 = SAI\_GROUP\_TYPE\_L2MC;

group\_api->create\_group(&l2mc\_group\_id, 1, attr);

## Create group members

sai\_object\_id\_t l2mc\_group\_member\_id1 = 0;

sai\_object\_id\_t l2mc\_group\_member\_id2 = 0;

attr[0].id = SAI\_GROUP\_MEMBER\_ATTR\_GROUP\_ID;

attr[0].value.oid = l2mc\_group\_id;

attr[1].id = SAI\_GROUP\_MEMBER\_ATTR\_MEMBER\_ID;

attr[1].value.oid = port1\_obj; /\* some valid port object\*/

group\_api->create\_group\_member (&l2mc\_group\_member\_id1, 2, attr);

attr[0].id = SAI\_GROUP\_MEMBER\_ATTR\_GROUP\_ID;

attr[0].value.oid = l2mc\_group\_id;

attr[1].id = SAI\_GROUP\_MEMBER\_ATTR\_MEMBER\_ID;

attr[1].value.oid = port2\_obj; /\* some valid port object\*/

group\_api->create\_group\_member(&l2mc\_group\_member\_id2, 2, attr);

## Create a multicast fdb entry associated with the group

sai\_api\_query(SAI\_API\_FDB, &sai\_fdb\_api);

sai\_fdb\_entry\_t fdb\_entry;

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

fdb\_entry.vlan\_id = 10;

fdb\_entry.mac\_address = {0x01, 0x00, 0x5e, 0x00, 0x02, 0x00}; /\* multicast\*/

attr[0].id = SAI\_FDB\_ENTRY\_ATTR\_TYPE;

attr[0].value.s32 = SAI\_FDB\_ENTRY\_STATIC;

attr[1].id = SAI\_FDB\_ENTRY\_ATTR\_PACKET\_ACTION;

attr[1].value.s32 = SAI\_PACKET\_ACTION\_FORWARD;

attr[2].id = SAI\_FDB\_ENTRY\_ATTR\_OUTPUT\_ID;

attr[2].value.obj = l2mc\_group\_id;

sai\_fdb\_api->sai\_create\_fdb\_entry\_fn(&fdb\_entry, 3, attr);