

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

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

**Contents**

[List of Changes i](#_Toc457504291)

[1 Overview 1](#_Toc457504292)

[2 Specification 1](#_Toc457504293)

[2.1 Changes to sai.h 1](#_Toc457504294)

[2.2 Change to sairouterinterface.h 1](#_Toc457504295)

[2.3 Change to sainexthopgroup.h 1](#_Toc457504296)

[2.4 New definitions in saiipmc.h 1](#_Toc457504297)

[2.5 Change to saihostif.h 4](#_Toc457504298)

[3 Examples 4](#_Toc457504299)

[3.1 Create nexthop group 4](#_Toc457504300)

[3.2 Create IPMC entry 4](#_Toc457504301)

[4 Appendix (review meeting questions) 5](#_Toc457504302)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| Initial Version | Base version |  | 17/10/2015 |
| 0.6 | Redesign | Min Yao | 17/10/2015 |
| 0.7 | Add RPF interface attributes in IPMC entry attributes | Min Yao | 25/8/2015 |

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

IP Multicast provides a solution to eliminate traffic redundancy and reduce server and CPU loads. This document covers details of IP multicast object –IP multicast group creating, IP multicast group removing, IP multicast group attributes setting and getting, RPF interface adding to IP multicast group, and RPF interface removing from IP multicast group.

# Specification

[This section describes the details of the proposed interface/API]

## Changes to sai.h

/\*

typedef enum \_sai\_api\_t {

SAI\_API\_IPMC= 45, /\* sai\_ipmc\_api\_t \*/

} sai\_api\_t;

## Change to sairouterinterface.h

/\*\*

\* @brief Routing interface attribute IDs

\*/

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

{

/\*\* READ-WRITE \*/

**/\*\* V4 Mcast enable [bool] (CREATE\_AND\_SET) (default to TRUE) \*/**

**SAI\_ROUTER\_INTERFACE\_ATTR\_V4\_MCAST\_ENABLE,**

**/\*\* V6 Mcast enable [bool] (CREATE\_AND\_SET) (default to TRUE) \*/**

**SAI\_ROUTER\_INTERFACE\_ATTR\_V6\_MCAST\_ENABLE,**

} sai\_router\_interface\_attr\_t;

## Change to saigroup.h

/\*\*

\* @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,

**/\*\* Group members are outputs of a L3 multicast group \*/**

**SAI\_GROUP\_TYPE\_IPMC,**

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

} sai\_group\_type\_t;

## New definitions in saiipmc.h

/\*\*

\* @brief Enum defining ipmc entry types.

\*/

typedef enum \_sai\_ipmc\_entry\_type\_t

{

/\*\* IPMC entry with type (S,G) \*/

SAI\_IPMC\_TYPE\_SG = 0x00000001,

/\*\* IPMC entry with type (\*,G) \*/

SAI\_IPMC\_TYPE\_XG = 0x00000002,

} sai\_ipmc\_entry\_type\_t;

/\*

\* IP multicast entry key

\*/

typedef struct \_sai\_ipmc\_entry\_t

{

sai\_object\_id\_t vrf\_id;

sai\_ipmc\_entry\_type\_t type;

sai\_ip\_address\_t destination;

sai\_ip\_address\_t source;

} sai\_ipmc\_entry\_t;

/\*

\* Attribute Id for IP multicast entry

\*/

typedef enum \_sai\_ipmc\_entry\_attr\_t

{

SAI\_IPMC\_ENTRY\_ATTR\_START,

/\*\* READ-ONLY \*/

/\*\* READ-WRITE \*/

**/\*\* RPF router interface list [sai\_object\_list\_t] (CREATE\_AND\_SET)**

**\* Empty list indicates that RPF interface checking is**

**\* disabled for the multicast group \*/**

**SAI\_IPMC\_ATTR\_RPF\_ROUTER\_INTERFACE\_LIST,**

/\*\* Packet action [sai\_packet\_action\_t]

(default to SAI\_PACKET\_ACTION\_FORWARD) \*/

SAI\_IPMC\_ATTR\_PACKET\_ACTION = SAI\_IPMC\_ENTRY\_ATTR\_START,

/\*\* Packet priority for trap/log actions [uint8\_t]

(default to 0) \*/

SAI\_IPMC\_ATTR\_TRAP\_PRIORITY,

/\*\* Output group id for the packet [sai\_object\_id\_t]

\* The generic group type should be SAI\_GROUP\_TYPE\_IPMC

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

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

SAI\_IPMC\_ATTR\_OUTPUT\_GROUP\_ID,

/\* Custom range base value \*/

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

} sai\_ipmc\_entry\_attr\_t;

/\*

\* Routine Description:

\* Create IP multicast entry

\*

\* Arguments:

\* [in] ipmc\_entry - IP multicast entry

\* [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\_ipmc\_entry\_fn)(

\_In\_ const sai\_ipmc\_entry\_t\* ipmc\_entry,

\_In\_ uint32\_t attr\_count,

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

);

/\*

\* Routine Description:

\* Remove IP multicast entry

\*

\* Arguments:

\* [in] ipmc\_entry - IP multicast entry

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_ipmc\_entry\_fn)(

\_In\_ const sai\_ipmc\_entry\_t\* ipmc\_entry

);

/\*

\* Routine Description:

\* Set IP multicast entry attribute value

\*

\* Arguments:

\* [in] IP multicast - IP multicast entry

\* [in] attr - attribute

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_ipmc\_entry\_attribute\_fn)(

\_In\_ const sai\_ipmc\_entry\_t\* ipmc\_entry,

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

);

/\*

\* Routine Description:

\* Get IP multicast entry attribute value

\*

\* Arguments:

\* [in] ipmc\_entry - IP multicast entry

\* [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\_ipmc\_entry\_attribute\_fn)(

\_In\_ const sai\_ipmc\_entry\_t\* ipmc\_entry,

\_In\_ uint32\_t attr\_count,

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

);

/\*

\* IP multicast method table retrieved with sai\_api\_query()

\*/

typedef struct \_sai\_ipmc\_api\_t

{

sai\_create\_ipmc\_entry\_fn create\_ipmc\_entry;

sai\_remove\_ipmc\_entry\_fn remove\_ipmc\_entry;

sai\_set\_ipmc\_entry\_attribute\_fn set\_ipmc\_entry\_attribute;

sai\_get\_ipmc\_entry\_attribute\_fn get\_ipmc\_entry\_attribute;

} sai\_ipmc\_api\_t;

## Change to sairouter.h

typedef enum \_sai\_virtual\_router\_attr\_t

{

SAI\_VIRTUAL\_ROUTER\_ATTR\_START,

/\*\* READ-WRITE \*/

**/\*\* Action for Unknown L3 multicast packets [sai\_packet\_action\_t]**

**\* (default to SAI\_PACKET\_ACTION\_DROP) \*/**

**SAI\_VIRTUAL\_ROUTER\_ATTR\_UNKNOWN\_L3\_MULTICAST\_PACKET\_ACTION,**

} sai\_virtual\_router\_attr\_t;

## Change to saihostif.h

typedef enum \_sai\_hostif\_trap\_id\_t

{

**/\*\* unknown l3 multicast packets**

**\* (default packet action is drop) \*/**

**SAI\_HOSTIF\_TRAP\_ID\_UNKNOWN\_L3\_MULTICAST,**

SAI\_HOSTIF\_TRAP\_ID\_CUSTOM\_EXCEPTION\_RANGE\_BASE = 0x00005000,

} sai\_hostif\_trap\_id\_t;

# Examples

Router interface should be created first, for example, rif1\_object, rif2\_object and rpf\_object have been created.

## Create a generic group

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

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

sai\_attribute\_t attr[1] = {0};

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

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

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

## Create group members

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

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

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

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

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

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

attr[1].value.oid = routerinterface\_obj1; /\* some valid routerinterface object\*/

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

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

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

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

attr[1].value.oid = routerinterface\_obj2; /\* some valid routerinterface object\*/

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

## Create IPMC entry

sai\_api\_query(SAI\_API\_IPMC, &sai\_ipmc\_api);

sai\_ipmc\_entry\_t ipmc\_entry;

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

sai\_object\_id\_t rpf\_interface\_list\_obj[2] = {0};

ipmc\_entry.vrf\_id = vr\_id\_obj;

ipmc\_entry.type = SAI\_IPMC\_TYPE\_XG;

ipmc\_entry.destination.addr\_family = SAI\_IP\_ADDR\_FAMILY\_IPV4;

ipmc\_entry.destination.addr.ip4 = 0xE0010101;

attr[0].id = SAI\_IPMC\_ATTR\_PACKET\_ACTION;

attr[0].value.s32 = SAI\_PACKET\_ACTION\_FORWARD;

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

attr[1].value.oid = ipmc\_group\_id;

rpf\_interface\_list\_obj[0] = routerinterface\_object;

attr[2].id = SAI\_IPMC\_ATTR\_RPF\_ROUTER\_INTERFACE\_LIST;

attr[2].value.objlist.count = 1;

attr[2].value.objlist.list = rpf\_interface\_list\_obj;

sai\_ipmc\_api->create\_ipmc\_entry(&ipmc\_entry, 3, attr);