

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **MPLS** |
| **Authors** | **Mellanox** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **05/03/2017** |
| **SAI-Version** | **1.1.0** |

**Contents**

[List of Changes i](#_Toc478055070)

[1 Overview 1](#_Toc478055071)

[1.1 InSegment 1](#_Toc478055072)

[1.2 OutSegment(SAI next hop) 1](#_Toc478055073)

[1.3 LabelStack 2](#_Toc478055074)

[1.4 Configuration examples 2](#_Toc478055075)

[*1.4.1* *LSR flow* 2](#_Toc478055076)

[1.4.2 Ingress LER 2](#_Toc478055077)

[1.4.3 Egress LER 2](#_Toc478055078)

[2 Pipeline model 2](#_Toc478055079)

[2.1 LSR flow 2](#_Toc478055080)

[2.2 Ingress LER flow 2](#_Toc478055081)

[2.3 Egress LER flow 2](#_Toc478055082)

[2.4 Egress LER PHP flow 3](#_Toc478055083)

[2.5 L3VPN 3](#_Toc478055084)

[2.6 Segment routing 3](#_Toc478055085)

[3 Specification 3](#_Toc478055086)

[3.1 SAI object 3](#_Toc478055087)

[3.2 SAI router interface 4](#_Toc478055088)

[3.3 SAI next hop 7](#_Toc478055089)

[3.4 SAI InSegment 8](#_Toc478055090)

[3.5 MPLS API summary 11](#_Toc478055091)

[4 Examples 11](#_Toc478055092)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 0.9.2 | Base version |  | 2/11/2015 |
| 1.1.0 | Update |  | 05/03/2017 |

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

MPLS library provides an access to MPLS functionality of the switch. SAI provides tools to configure the switch to forward MPLS packets, create basic MPLS objects in HW, bind them together in order to create MPLS flow in HW. Also SAI provides ability to protect MPLS LSPs, terminate MPLS LSP and forward packets to IP router and encapsulate IP packets with MPLS.

Scope : IP MPLS MLPs tunnel will be added at latter stage

Three basic objects are defined in order to manage MPLS functionality:

- InSegment

- OutSegment

- LabelStack

## InSegment

An InSegment defines incoming MPLS segment to a Label Switch Router (LSR), needed parameters and actions. This object represents an incoming label as it was created by signaling protocol, network management or node administrator and POP operation of the LSR.

The InSegment contains following parameters:

* Incoming label
* Number of pops
* Action
* Counter

InSegment can be bound to LabelStack or OutSegment in order to create a LSR flow in HW.

InSegment can be bound to multiple objects to create ECMP.

InSegment can be a termination point and cause packet re-lookup.

## OutSegment(SAI next hop)

An OutSegment defines outgoing parameters from Label Switch Router to a MPLS network. This object represents outgoing top MPLS label. This object performs PUSH and PHP operations.

OutSegment has following parameters:

- labelStack object

- Next IP hop

OutSegment can be bound to InSegment LSR flow , and can be bound to route in in order to create ingress LER .

Several InSegments or LabelSteacks can be bound to same OutSegment to create goto-tunnel flow.

## LabelStack

A LabelStack defines a list of labels that are pushed to the packet beneath top label defined by OutSegment.

The LabelStack is a group of labels that can be bound to another objects in order to create LSR path or backup path.

## Configuration examples

### *LSR flow*

Label stack

In segment

Out segment

### Ingress LER

Router

Label stack

Out segment

### Egress LER

In segment

Out segment

# Pipeline model

## LSR flow



## Ingress LER flow



## Egress LER flow



## Egress LER PHP flow



## L3VPN



## Segment routing



# Specification

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

## SAI object

/\*\*

\* @brief Structure for bulk retrieval of objectids, attribute and values for

\* each object-type. Key will be used in case of object-types not having

\* object-ids.

\*/

typedef struct \_sai\_object\_key\_t

{

union

{

sai\_object\_id\_t object\_id;

sai\_fdb\_entry\_t fdb\_entry;

sai\_neighbor\_entry\_t neighbor\_entry;

sai\_route\_entry\_t route\_entry;

sai\_mcast\_fdb\_entry\_t mcast\_fdb\_entry;

sai\_l2mc\_entry\_t l2mc\_entry;

sai\_ipmc\_entry\_t ipmc\_entry;

**sai\_in\_segment\_entry inseg\_entry;**

} key;

} sai\_object\_key\_t;

## SAI router interface

typedef enum \_sai\_router\_interface\_type\_t

{

/\*\* Port or Lag Router Interface Type \*/

SAI\_ROUTER\_INTERFACE\_TYPE\_PORT,

/\*\* VLAN Router Interface Type \*/

SAI\_ROUTER\_INTERFACE\_TYPE\_VLAN,

/\*\* Loopback Router Interface Type \*/

SAI\_ROUTER\_INTERFACE\_TYPE\_LOOPBACK,

/\*\* Loopback Router Interface Type \*/

**SAI\_ROUTER\_INTERFACE\_TYPE\_MPLS\_ROUTER**

/\*\* Sub port Router Interface Type \*/

SAI\_ROUTER\_INTERFACE\_TYPE\_SUB\_PORT,

/\*\* .1D Bridge Router Interface Type \*/

SAI\_ROUTER\_INTERFACE\_TYPE\_BRIDGE

} sai\_router\_interface\_type\_t;

/\*\*

\* @brief Routing interface attribute IDs

\*/

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

{

/\*\*

\* @brief Start of attributes

\*/

SAI\_ROUTER\_INTERFACE\_ATTR\_START,

/\* READ-ONLY \*/

/\*\*

\* @brief Virtual router id

\*

\* @type sai\_object\_id\_t

\* @flags MANDATORY\_ON\_CREATE | CREATE\_ONLY

\* @objects SAI\_OBJECT\_TYPE\_VIRTUAL\_ROUTER

\*/

SAI\_ROUTER\_INTERFACE\_ATTR\_VIRTUAL\_ROUTER\_ID = SAI\_ROUTER\_INTERFACE\_ATTR\_START,

/\*\*

\* @brief Type

\*

\* @type sai\_router\_interface\_type\_t

\* @flags MANDATORY\_ON\_CREATE | CREATE\_ONLY

\*/

SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE,

/\*\*

\* @brief Assosiated Port or Lag object id

\*

\* @type sai\_object\_id\_t

\* @flags MANDATORY\_ON\_CREATE | CREATE\_ONLY

\* @objects SAI\_OBJECT\_TYPE\_PORT, SAI\_OBJECT\_TYPE\_LAG

\* @condition SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE == SAI\_ROUTER\_INTERFACE\_TYPE\_PORT or SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE == SAI\_ROUTER\_INTERFACE\_TYPE\_SUB\_PORT

\*/

SAI\_ROUTER\_INTERFACE\_ATTR\_PORT\_ID,

/\*\*

\* @brief Assosiated Vlan

\*

\* @type sai\_object\_id\_t

\* @flags MANDATORY\_ON\_CREATE | CREATE\_ONLY

\* @objects SAI\_OBJECT\_TYPE\_VLAN

\* @condition SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE == SAI\_ROUTER\_INTERFACE\_TYPE\_VLAN or SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE == SAI\_ROUTER\_INTERFACE\_TYPE\_SUB\_PORT

\*/

SAI\_ROUTER\_INTERFACE\_ATTR\_VLAN\_ID,

/\* READ-WRITE \*/

/\*\*

\* @brief MAC Address

\*

\* Not valid when #SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE == #SAI\_ROUTER\_INTERFACE\_TYPE\_LOOPBACK)

\* Default to #SAI\_VIRTUAL\_ROUTER\_ATTR\_SRC\_MAC\_ADDRESS if not set on create)

\*

\* @type sai\_mac\_t

\* @flags CREATE\_AND\_SET

\* @default attrvalue SAI\_VIRTUAL\_ROUTER\_ATTR\_SRC\_MAC\_ADDRESS

\*/

SAI\_ROUTER\_INTERFACE\_ATTR\_SRC\_MAC\_ADDRESS,

/\*\*

\* @brief Admin V4 state

\*

\* @type bool

\* @flags CREATE\_AND\_SET

\* @default true

\*/

/\*\*

\* @brief Assosiated Port or Lag object id

\*

\* @type sai\_object\_id\_t

\* @flags MANDATORY\_ON\_CREATE | CREATE\_ONLY

\* @objects SAI\_OBJECT\_TYPE\_RIF,

\* @condition SAI\_ROUTER\_INTERFACE\_ATTR\_TYPE == SAI\_ROUTER\_INTERFACE\_TYPE\_MPLS\_ROUTER \*/

SAI\_ROUTER\_INTERFACE\_ATTR\_RIF\_ID,

SAI\_ROUTER\_INTERFACE\_ATTR\_ADMIN\_V4\_STATE,

/\*\*

\* @brief Admin V6 state

\*

\* @type bool

\* @flags CREATE\_AND\_SET

\* @default true

\*/

SAI\_ROUTER\_INTERFACE\_ATTR\_ADMIN\_V6\_STATE,

/\*\*

\* @brief MTU

\*

\* @type sai\_uint32\_t

\* @flags CREATE\_AND\_SET

\* @default 1514

\*/

SAI\_ROUTER\_INTERFACE\_ATTR\_MTU,

/\*\*

\* @brief RIF bind point for ingress ACL object

\*

\* Bind (or unbind) an ingress acl table or acl group on a RIF. Enable/Update

\* ingress ACL table or ACL group filtering by assigning a valid object id.

\* Disable ingress filtering by assigning SAI\_NULL\_OBJECT\_ID in the

\* attribute value.

\*

\* @type sai\_object\_id\_t

\* @flags CREATE\_AND\_SET

\* @objects SAI\_OBJECT\_TYPE\_ACL\_TABLE, SAI\_OBJECT\_TYPE\_ACL\_TABLE\_GROUP

\* @allownull true

\* @default SAI\_NULL\_OBJECT\_ID

\*/

SAI\_ROUTER\_INTERFACE\_ATTR\_INGRESS\_ACL,

/\*\*

\* @brief RIF bind point for egress ACL object

\*

\* Bind (or unbind) an egress acl table or acl group on a RIF. Enable/Update

\* egress ACL table or ACL group filtering by assigning a valid object id.

\* Disable egress filtering by assigning SAI\_NULL\_OBJECT\_ID

\* in the attribute value.

\*

\* @type sai\_object\_id\_t

\* @flags CREATE\_AND\_SET

\* @objects SAI\_OBJECT\_TYPE\_ACL\_TABLE, SAI\_OBJECT\_TYPE\_ACL\_TABLE\_GROUP

\* @allownull true

\* @default SAI\_NULL\_OBJECT\_ID

\*/

SAI\_ROUTER\_INTERFACE\_ATTR\_EGRESS\_ACL,

/\*\*

\* @brief Packet action when neighbor table lookup miss for this router interface

\*

\* @type sai\_packet\_action\_t

\* @flags CREATE\_AND\_SET

\* @default SAI\_PACKET\_ACTION\_TRAP

\*/

SAI\_ROUTER\_INTERFACE\_ATTR\_NEIGHBOR\_MISS\_PACKET\_ACTION,

/\*\*

\* @brief V4 Mcast enable

\*

\* @type bool

\* @flags CREATE\_AND\_SET

\* @default false

\*/

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

/\*\*

\* @brief V6 Mcast enable

\*

\* @type bool

\* @flags CREATE\_AND\_SET

\* @default false

\*/

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

/\*\*

\* @brief End of attributes

\*/

SAI\_ROUTER\_INTERFACE\_ATTR\_END,

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

SAI\_ROUTER\_INTERFACE\_ATTR\_CUSTOM\_RANGE\_START = 0x10000000,

/\*\* End of custom range base \*/

SAI\_ROUTER\_INTERFACE\_ATTR\_CUSTOM\_RANGE\_END

} sai\_router\_interface\_attr\_t;

## SAI next hop

typedef enum \_sai\_next\_hop\_type\_t

{

/\*\* IP next hop \*/

SAI\_NEXT\_HOP\_TYPE\_IP,

**/\*\* MPLS(outsegment) next hop \*/**

**SAI\_NEXT\_HOP\_TYPE\_MPLS,**

/\*\* Tunnel next hop \*/

SAI\_NEXT\_HOP\_TYPE\_TUNNEL\_ENCAP

} sai\_next\_hop\_type\_t;

/\*\*

\* @brief Attribute id for next hop

\*/

typedef enum \_sai\_next\_hop\_attr\_t

{

/\*\*

\* @brief Start of attributes

\*/

SAI\_NEXT\_HOP\_ATTR\_START,

/\*\*

\* @brief Next hop entry type

\*

\* @type sai\_next\_hop\_type\_t

\* @flags MANDATORY\_ON\_CREATE | CREATE\_ONLY

\*/

SAI\_NEXT\_HOP\_ATTR\_TYPE = SAI\_NEXT\_HOP\_ATTR\_START,

/\*\*

\* @brief Next hop entry ipv4 address

\*

\* @type sai\_ip\_address\_t

\* @flags MANDATORY\_ON\_CREATE | CREATE\_ONLY

\* @condition SAI\_NEXT\_HOP\_ATTR\_TYPE == SAI\_NEXT\_HOP\_TYPE\_IP

\*/

SAI\_NEXT\_HOP\_ATTR\_IP,

/\*\*

\* @brief Next hop entry router interface id

\*

\* @type sai\_object\_id\_t

\* @flags MANDATORY\_ON\_CREATE | CREATE\_ONLY

\* @objects SAI\_OBJECT\_TYPE\_ROUTER\_INTERFACE

\*/

SAI\_NEXT\_HOP\_ATTR\_ROUTER\_INTERFACE\_ID,

/\*\*

\* @brief Next hop entry tunnel-id

\*

\* @type sai\_object\_id\_t

\* @flags MANDATORY\_ON\_CREATE | CREATE\_ONLY

\* @objects SAI\_OBJECT\_TYPE\_TUNNEL

\* @condition SAI\_NEXT\_HOP\_ATTR\_TYPE == SAI\_NEXT\_HOP\_TYPE\_TUNNEL\_ENCAP

\*/

SAI\_NEXT\_HOP\_ATTR\_TUNNEL\_ID,

**/\*\***

**\* @brief push label**

**\***

**\* @type sai\_object\_list\_t**

**\* @flags MANDATORY\_ON\_CREATE | CREATE\_ONLY**

**\* @condition SAI\_NEXT\_HOP\_ATTR\_TYPE == SAI\_NEXT\_HOP\_TYPE\_MPLS**

**\*/**

**SAI\_NEXT\_HOP\_ATTR\_LABELSTACK,**

/\*\*

\* @brief End of attributes

\*/

SAI\_NEXT\_HOP\_ATTR\_END,

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

SAI\_NEXT\_HOP\_ATTR\_CUSTOM\_RANGE\_START = 0x10000000,

/\*\* End of custom range base \*/

SAI\_NEXT\_HOP\_ATTR\_CUSTOM\_RANGE\_END

} sai\_next\_hop\_attr\_t;

## SAI InSegment

/\*

\* in\_segment entry

\*/

typedef struct \_sai\_in\_segment\_entry\_t

{

SAI\_INSEG\_ENTRY\_ATTR\_STAR;

/\*\* incammind label \*/

SAI\_INSEG\_ENTRY\_INCOMING\_LABL,

/\*\* number of pop \*/

SAI\_INSEG\_ENTRY\_NUM\_OF\_PO,

/\*\*

\* @brief Packet action

\* @type sai\_packet\_action\_t

\* @flags CREATE\_AND\_SET

\* @default SAI\_PACKET\_ACTION\_FORWARD

\*/

SAI\_ INSEG \_ENTRY\_ATTR\_PACKET\_ACTION,

/\*\*

\* @brief Packet priority for trap/log actions

\* @type sai\_uint8\_t

\* @flags CREATE\_AND\_SET

\* @default 0

\*/

SAI\_INSEG\_ENTRY\_ATTR\_TRAP\_PRIORITY,

/\*\*

\* @brief The next hop id

\* @type sai\_object\_id\_t

\* @flags CREATE\_AND\_SET

\* @default SAI\_NULL\_OBJECT\_ID

\*/

SAI\_INSEG\_ENTRY\_ATTR\_NEXT\_HOP\_ID,

SAI\_INSEG\_ENTRY\_ATTR\_CUSTOM\_RANGE\_START,

SAI\_ROUTE\_ENTRY\_ATTR\_CUSTOM\_RANGE\_END

} sai\_in\_segment\_entry\_t;

/\*

\* Routine Description:

\* Create in\_segment entry

\*

\* Arguments:

\* [in] in\_segment\_entry - InSegment entry

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

\* [in] attrs - array of attributes

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*

\*

\*/

typedef sai\_status\_t (\*sai\_create\_in\_segment\_entry\_fn)(

\_In\_ const sai\_in\_segment\_entry\_t\* in\_segment\_entry,

\_In\_ uint32\_t attr\_count,

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

);

/\*

\* Routine Description:

\* Remove in\_segment entry

\*

\* Arguments:

\* [in] in\_segment\_entry - InSegment entry

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*

\*/

typedef sai\_status\_t (\*sai\_remove\_in\_segment\_entry\_fn)(

\_In\_ const sai\_in\_segment\_entry\_t\* in\_segment\_entry

);

/\*

\* Routine Description:

\* Set in\_segment entry attribute value

\*

\* Arguments:

\* [in] in\_segment\_entry - InSegment entry

\* [in] attr - attribute

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

\_In\_ const sai\_in\_segment\_entry\_t\* in\_segment\_entry,

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

);

/\*

\* Routine Description:

\* Get in\_segment entry attribute value

\*

\* Arguments:

\* [in] in\_segment\_entry - InSegment entry

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

\* [inout] attrs - array of attributes

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_get\_in\_segment\_entry\_attribute\_fn)(

\_In\_ const sai\_in\_segment\_entry\_t\* in\_segment\_entry,

\_In\_ uint32\_t attr\_count,

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

);

typedef sai\_status\_t (\*sai\_get\_out\_segment\_entry\_attribute\_fn)(

\_In\_ const sai\_out\_segment\_id\_t\* out\_segment\_entry,

\_In\_ uint32\_t attr\_count,

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

);

## MPLS API summary

/\*

\* MPLS method table retrieved with sai\_api\_query()

\*/

typedef struct \_sai\_mpls\_api\_t

{

sai\_create\_in\_segment\_entry\_fn create\_in\_segment\_entry;

sai\_remove\_in\_segment\_entry\_fn remove\_in\_segment\_entry;

sai\_set\_in\_segment\_entry\_attribute\_fn set\_in\_segment\_entry\_attribute;

sai\_get\_in\_segment\_entry\_attribute\_fn get\_in\_segment\_entry\_attribute;

} sai\_mpls\_api\_t;

# Examples