

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **MPLS** |
| **Authors** | **Mellanox** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **2/11/2015** |
| **SAI-Version** | **0.9.2** |

**Contents**

[List of Changes i](#_Toc412068622)

[1 Overview 1](#_Toc412068623)

[1.1 InSegment 1](#_Toc412068624)

[1.2 OutSegment 1](#_Toc412068625)

[1.3 LabelStack 1](#_Toc412068626)

[1.4 Configuration examples 2](#_Toc412068627)

[1.4.1 LSR flow 2](#_Toc412068628)

[1.4.2 Ingress LER 2](#_Toc412068629)

[1.4.3 Egress LER 2](#_Toc412068630)

[2 Specification 2](#_Toc412068631)

[2.1 MPLS attributes 2](#_Toc412068632)

[2.2 ILM table 3](#_Toc412068633)

[2.3 InSegment functions 4](#_Toc412068634)

[2.4 OutSegment functions 6](#_Toc412068635)

[2.5 LabelStack functions 7](#_Toc412068636)

[2.6 MPLS API summary 8](#_Toc412068637)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 0.9.2 | Base version |  | 2/11/2015 |

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

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.

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

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:

- Top label

- Next IP hop

OutSegment can be bound to InSegment or InSegment and LabelStack in order to create LSR flow and can be bound to another OutSegment in order to create a backup path.

Several InSegments or LabelStacks 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 linked list of labels that can be bound to another objects in order to create LSR path or backup path.

## Configuration examples

### LSR flow



An InSegment is connected to LabelStack and OutSegment or OutSegment only to create basic LSR flow

If InSegment or LabelStack points to multiple objects in parallel – it creates an ECMP flow.

### Ingress LER



A LabelStack or OutSegment or link list of them can be pointed from IP route in order to create Ingress LER flow. If single route points to multiple objects in parallel, it creates an ECMP flow.

### Egress LER



An InSegment can terminate current MPLS tunnel and command packet to perform another MPLS lookup or forward it to IP router.

# Specification

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

## MPLS attributes

/\*

\* Attribute Id for sai mpls objects

\*/

typedef enum \_sai\_mpls\_attr\_t

{

/\* READ-ONLY \*/

/\* READ-WRITE \*/

/\* MPLS entry packet action [sai\_packet\_action\_t] (default to SAI\_PACKET\_ACTION\_DROP) \*/

SAI\_MPLS\_ATTR\_PACKET\_ACTION,

/\* router interface id [sai\_router\_interface\_id\_t] (MANDATORY\_ON\_CREATE) \*/

SAI\_MPLS\_ATTR\_ROUTER\_INTERFACE\_ID,

/\* Protection object for LSP [sai\_mpls\_fwd\_container\_id\_t] \*/

SAI\_MPLS\_ATTR\_FWD\_CONTAINER,

/\* Top lavel of MPLS packet [sai\_label\_t] \*/

SAI\_MPLS\_ATTR\_LABEL,

/\* Next hop group id for the packet[sai\_next\_hop\_id\_t] \*/

SAI\_MPLS\_ATTR\_NEXT\_HOP\_ID,

/\* Protection object for LSP [sai\_mpls\_fwd\_container\_id\_t] \*/

SAI\_MPLS\_ATTR\_BACKUP\_TUNNEL\_ID

} sai\_mpls\_attr\_t;

## ILM table

/\*

\* Routine Description:

\* Create ILM table

\*

\* Arguments:

\* [out] ilm\_table\_id - ILM table id

\* [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\_ilm\_table\_fn)(

\_Out\_ sai\_ilm\_table\_id\_t\* ilm\_table\_id,

\_In\_ uint32\_t attr\_count,

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

);

/\*

\* Routine Description:

\* Remove ILM table

\*

\* Arguments:

\* [in] ilm\_table\_id - ILM table id

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*

\*/

typedef sai\_status\_t (\*sai\_remove\_ilm\_table\_fn)(

\_In\_ const sai\_ilm\_table\_id\_t ilm\_table\_id

);

/\*

\* Routine Description:

\* Set ILM table attribute value

\*

\* Arguments:

\* [in] ilm\_table\_id - ILM table id

\* [in] attr - attribute

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_ilm\_table\_attribute\_fn)(

\_In\_ const sai\_ilm\_table\_id\_t ilm\_table\_id,

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

);

/\*

\* Routine Description:

\* Get ILM table attribute value

\*

\* Arguments:

\* [in] ilm\_table\_id - ILM table id

\* [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\_ilm\_table\_attribute\_fn)(

\_In\_ const sai\_ilm\_table\_id\_t ilm\_table\_id,

\_In\_ uint32\_t attr\_count,

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

);

## InSegment functions

/\*

\* in\_segment entry

\*/

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

{

sai\_ilm\_table\_id\_t ilm\_table; /\* ILM table to create in\_segment \*/

sai\_label\_t incoming\_label; /\* incoming label value \*/

} 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

);

## OutSegment functions

/\*

\* Routine Description:

\* Create out\_segment entry

\*

\* Arguments:

\* [in] out\_segment\_entry - OutSegment 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\_out\_segment\_entry\_fn)(

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

\_In\_ uint32\_t attr\_count,

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

);

/\*

\* Routine Description:

\* Remove out\_segment entry

\*

\* Arguments:

\* [in] out\_segment\_entry - OutSegment entry

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*

\*/

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

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

);

/\*

\* Routine Description:

\* Set out\_segment entry attribute value

\*

\* Arguments:

\* [in] out\_segment\_entry - OutSegment entry

\* [in] attr - attribute

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

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

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

);

/\*

\* Routine Description:

\* Get out\_segment entry attribute value

\*

\* Arguments:

\* [in] out\_segment\_entry - OutSegment 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\_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

);

## LabelStack functions

/\*

\* Routine Description:

\* Create label\_stack entry

\*

\* Arguments:

\* [in] label\_stack\_entry - LabelStack 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\_label\_stack\_entry\_fn)(

\_In\_ const sai\_label\_stack\_id\_t\* label\_stack\_entry,

\_In\_ uint32\_t attr\_count,

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

);

/\*

\* Routine Description:

\* Remove label\_stack entry

\*

\* Arguments:

\* [in] label\_stack\_entry - LabelStack entry

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*

\*/

typedef sai\_status\_t (\*sai\_remove\_label\_stack\_entry\_fn)(

\_In\_ const sai\_label\_stack\_id\_t\* label\_stack\_entry

);

/\*

\* Routine Description:

\* Set label\_stack entry attribute value

\*

\* Arguments:

\* [in] label\_stack\_entry - LabelStack entry

\* [in] attr - attribute

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

\_In\_ const sai\_label\_stack\_id\_t\* label\_stack\_entry,

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

);

/\*

\* Routine Description:

\* Get label\_stack entry attribute value

\*

\* Arguments:

\* [in] label\_stack\_entry - LabelStack 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\_label\_stack\_entry\_attribute\_fn)(

\_In\_ const sai\_label\_stack\_id\_t\* label\_stack\_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\_ilm\_table\_fn create\_ilm\_table;

sai\_remove\_ilm\_table\_fn remove\_ilm\_table;

sai\_set\_ilm\_table\_attribute\_fn set\_ilm\_table\_attribute;

sai\_get\_ilm\_table\_attribute\_fn get\_ilm\_table\_attribute;

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\_create\_out\_segment\_entry\_fn create\_out\_segment\_entry;

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

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

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

sai\_create\_label\_stack\_entry\_fn create\_label\_stack\_entry;

sai\_remove\_label\_stack\_entry\_fn remove\_label\_stack\_entry;

sai\_set\_label\_stack\_entry\_attribute\_fn set\_label\_stack\_entry\_attribute;

sai\_get\_label\_stack\_entry\_attribute\_fn get\_label\_stack\_entry\_attribute;

} sai\_mpls\_api\_t;