

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **New API for xSTP** |
| **Authors** | **DELL** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **03/04** |
| **SAI-Version** | **V0.9.2** |

**Contents**

[List of Changes i](#_Toc413347797)

[1 Overview 1](#_Toc413347798)

[2 Specification 1](#_Toc413347799)

[2.1 Create STP instance 1](#_Toc413347800)

[2.2 Delete STP instance 2](#_Toc413347801)

[2.3 Update stp state on stp instance of a port 2](#_Toc413347802)

[2.4 Retrieve stp state on stp instance of a port 3](#_Toc413347803)

[2.5 Get/Set Attribute for XSTP instance 3](#_Toc413347804)

[2.6 Method Table 4](#_Toc413347805)

[2.7 Removing existing STP related Attribute from Port API List 5](#_Toc413347806)

[2.8 Associating a VLAN to STP instance 5](#_Toc413347807)

[2.9 Retrieve list of VLANs mapped for a STP instance 6](#_Toc413347808)

[3 Appendix 6](#_Toc413347809)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 0.9.2 | Proposal for xSTP | 0 | 03/04/2015 |
| 0.9.2 | Version 2   * Incorporated feedbacks. Appendix section has details of the changes |  | 03/05/2015 |

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

This document contains the proposal for the STP API’s. These API’s are needed for MSTP/PVST support. The user can configure Spanning tree states as well as Spanning tree instances with the defined APIs. A VLAN is associated with a spanning tree instance and by default all VLANs are part of default spanning tree instances. User can also associate a different spanning tree instance to a VLAN.

The user can also set the default STP instance ID. SAI\_STP\_DEFAULT\_INST\_ID attribute is used to set default STP instance ID for which the VLANS that are not part of any STP instance should be mapped to. Following are the behavior with respect to default Instance ID.

1. Default STP instance would be initialized in NPU initialization. When default instance is set all VLANs that are not part of any STP instance are mapped to default instance.
2. If a VLAN is added to any non-default STP instance then it is removed from default STP instance.
3. If a VLAN is removed from any non-default STP instance then its added back to default STP instance.
4. The default STP instance can be changed only through the attribute set calls. It cannot be externally created or removed.
5. VLANs cannot be explicitly added or removed from default STG instance. They would be manipulated internally based on addition or removal to other STG instances.

# Specification

**In sai.h**

typedef enum \_sai\_api\_t {

SAI\_API\_STP= 18, /\* sai\_stp\_api\_t \*/

}

**In saiswitch.h**

typedef enum \_sai\_switch\_attr\_t {

/\*Default SAI STP instance ID \*/

SAI\_SWITCH\_ATTR\_DEFAULT\_STP\_INST\_ID,

} sai\_switch\_attr\_t

**In saistatus.h**

/\*

\* Invalid STP Instance ID

\*/

#define SAI\_STATUS\_INVALID\_STP\_INSTANCE SAI\_STATUS\_CODE(0x00000018L)

**These changes are in new file saixstp.h**

## Create STP instance

/\*  
\* Routine Description:  
\*    Create a STP instance  
\*  
\* Arguments:  
\*    [out] inst\_id - Spanning tree instance identifier  
\*  
\* Return Values:  
\*    SAI\_STATUS\_SUCCESS on success  
\*    Failure status code on error  
\*/

typedef sai\_status\_t (\*sai\_create\_stp\_inst\_fn)  
  (  
     \_Out\_ sai\_stp\_instance\_id\_t \*inst\_id,     /\* instance id\*/  
  )

## Delete STP instance

/\*  
\* Routine Description:  
\*    Delete STP instance  
\*  
\* Arguments:  
\*    [in] inst\_id - STP instance identifier  
\*  
\* Return Values:  
\*    SAI\_STATUS\_SUCCESS on success  
\*    Failure status code on error  
\*/

typedef sai\_status\_t (\*sai\_remove\_stp\_inst\_fn)  
  (  
     \_In\_ sai\_stp\_instance\_id\_t inst\_id,     /\* instance id\*/  
  )

## Update stp state on stp instance of a port

/\*  
\* Routine Description:  
\*    Update stp state of a port in specified stp instance.    
\*  
\* Arguments:  
\*    [in] inst\_id - STP instance identifier  
\*    [in] port\_id - port identifier  
\*    [in] stp\_port\_state - STP state  
\*  
\* Return Values:  
\*    SAI\_STATUS\_SUCCESS on success  
\*    Failure status code on error  
\*/

typedef sai\_status\_t (\*sai\_set\_stp\_port\_state\_fn)  
 (  
     \_In\_ sai\_stp\_instance\_id\_t inst\_id,     /\* instance id\*/  
     \_In\_ sai\_port\_id\_t port\_id,   /\*port identifier\*/  
     \_In\_ sai\_port\_stp\_port\_state\_t  stp\_port\_state /\*STP state\*/  
 )

## Retrieve stp state on stp instance of a port

/\*  
\* Routine Description:  
\*    Get stp state of a port in specified stp instance.    
\*  
\* Arguments:  
\*    [in] inst\_id - STP instance identifier  
\*    [in] port\_id - port identifier  
\*    [out] stp\_port\_state - STP state  
\*  
\* Return Values:  
\*    SAI\_STATUS\_SUCCESS on success  
\*    Failure status code on error  
\*/

typedef sai\_status\_t (\*sai\_get\_stp\_port\_state\_fn)  
  (  
     \_In\_ sai\_stp\_instance\_id\_t inst\_id,     /\* instance id\*/  
     \_In\_ sai\_port\_id\_t port\_id,   /\*port identifier\*/  
     \_Out\_ sai\_port\_stp\_port\_state\_t  \*stp\_port\_state /\*STP state\*/  
  )

## Get/Set Attribute for XSTP instance

/\*

\* Routine Description:

\* Set STP attribute

\*

\* Arguments:

\* [in] sai\_stp\_instance\_id\_t – stp id

\* [in] attr - attribute

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

\_In\_ sai\_stp\_instance\_id\_t stp\_id,

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

);

/\*

\* Routine Description:

\* Get STP attribute

\*

\* Arguments:

\* [in] sai\_stp\_instance\_id\_t – stp 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\_stp\_attribute\_fn)(

\_In\_ sai\_stp\_instance\_id\_t stp\_id,

\_In\_ unsigned int attr\_count,

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

);

## Method Table

/\* \* STP methods table retrieved with sai\_api\_query() \*/

typedef struct \_sai\_xstp\_api\_t {

sai\_create\_stp\_inst\_fn  create\_stp\_inst;

sai\_remove\_stp\_inst\_fn remove\_stp\_inst;

sai\_set\_stp\_port\_state\_fn set\_stp\_port\_state;

sai\_get\_stp\_port\_state\_fn get\_stp\_port\_state;

sai\_set\_stp\_attribute\_fn set\_stp\_attribute;

sai\_get\_stp\_attribute\_fn get\_stp\_attribute;

} sai\_stp\_api\_t;

## Removing existing STP related Attribute from Port API List

**In saiport.h**

The below attribute needs to be removed from the list of port attributes

/\* Stp mode [sai\_port\_stp\_state\_t] \*/

SAI\_PORT\_ATTR\_STP\_STATE,

**The below definition of STP States needs to be moved from saiport.h to this the saixstp.h file**

/\*

\* Attribute data for SAI\_PORT\_STP\_STATE

\*/

typedef enum \_sai\_port\_stp\_state\_t

{

/\* Port is Learning \*/

SAI\_PORT\_STP\_STATE\_LEARNING,

/\* Port is Forwarding \*/

SAI\_PORT\_STP\_STATE\_FORWARDING,

/\* Port is Blocking \*/

SAI\_PORT\_STP\_STATE\_BLOCKING,

} sai\_port\_stp\_state\_t;

## Associating a VLAN to STP instance

**In saivlan.h**

Add an attribute in existing VLAN Module to associate a STP instance with a VLAN ID.

typedef enum \_sai\_vlan\_attr\_t

{

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

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

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

SAI\_VLAN\_ATTR\_STP\_INSTANCE,

/\* Custom range base value \*/

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

} sai\_vlan\_attr\_t;

## Retrieve list of VLANs mapped for a STP instance

Use SAI\_STP\_ATTR\_VLAN attribute. Use vlan\_data of type sai\_vlan\_list\_t to access the attribute

# Appendix

Review Comments – Atit 03/05/2015

**From:** Jain, Atit [<mailto:Atit.Jain@caviumnetworks.com>]   
**Sent:** Thursday, March 05, 2015 6:33 PM  
**To:** Manickam, Arunsubash; [opencompute-networking@lists.opencompute.org](mailto:opencompute-networking@lists.opencompute.org)  
**Subject:** RE: API Proposal for STG

Hi Arun,

The SAI\_STP\_ DEFAULT\_INST\_ID should be part of SAI Switch attributes because its per NPU/Switch default. In its current form, SAI STP this as an attribute is set for a sai\_stp\_instance\_id\_t stp\_id which is like setting default instance per instance.

Few minor comments are in the file attached.

Regards,  
Atit

**From:** [opencompute-networking-bounces@lists.opencompute.org](mailto:opencompute-networking-bounces@lists.opencompute.org) [<mailto:opencompute-networking-bounces@lists.opencompute.org>] **On Behalf Of** Manickam, Arunsubash  
**Sent:** Thursday, March 05, 2015 6:43 PM  
**To:** [Atit.Jain@caviumnetworks.com](mailto:Atit.Jain@caviumnetworks.com); [opencompute-networking@lists.opencompute.org](mailto:opencompute-networking@lists.opencompute.org)  
**Subject:** Re: [Opencompute-networking] API Proposal for STG

Hi Atit,

Thanks for your quick feedback

1 – I agree that SAI\_STP\_DEFAULT\_INST\_ID should be part of switch attribute. This would be inline with default VLAN ID which is a switch attribute - SAI\_SWITCH\_ATTR\_DEFAULT\_PORT\_VLAN\_ID

2 – Though we will have no attributes defined we will have the get/set attribute functions defined

3 - SAI\_STATUS\_INVALID\_STP\_INSTANCE – Yes we will add this new error code to return failure for an invalid instance passed

We will update this and send an updated proposal

Thanks,

Arun