

|  |
| --- |
|  |

802.1BR Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **802.1BR** |
| **Authors** | **DELL** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **11/06/2015** |
| **SAI-Version** | **V0.9.3** |

**Contents**

[List of Changes i](#_Toc461043477)

[1 Overview 1](#_Toc461043478)

[2 Overview 2](#_Toc461043479)

[2.1 802.1BR Introduction 2](#_Toc461043480)

[2.1.1 Forwarding Tables at CB 3](#_Toc461043481)

[2.1.2 Forwarding Table at PE1 4](#_Toc461043482)

[2.1.3 Forwarding Table at PE2 4](#_Toc461043483)

[2.2 Lag of Extended Ports 4](#_Toc461043484)

[2.3 xSTP on Extended Ports 6](#_Toc461043485)

[3 Dot1br v/s Tunnel model 7](#_Toc461043486)

[5 Specification 8](#_Toc461043487)

[5.1 Changes to saitypes.h 8](#_Toc461043488)

[5.2 Changes to sai.h 8](#_Toc461043489)

[5.3 Changes to saiport.h 8](#_Toc461043490)

[5.4 Changes to saifdb.h 8](#_Toc461043491)

[5.5 New File saidot1brport.h 9](#_Toc461043492)

[5.6 New File saidot1brextport.h 11](#_Toc461043493)

[5.7 Changes to saivlan.h 14](#_Toc461043494)

[5.8 New file saidot1brecidfwd.h 15](#_Toc461043495)

[6 Configuration Example 18](#_Toc461043496)

[6.1 Creating and Deleting an Extended Port 18](#_Toc461043497)

[6.2 Vlan configuration 18](#_Toc461043498)

[6.2.1 Adding extended ports to the Vlan 18](#_Toc461043499)

[6.2.2 Assigning Flooding ECID to the Vlan 19](#_Toc461043500)

[6.2.3 Removing extended ports from the Vlan 19](#_Toc461043501)

[6.3 Creating/Deleting 802.1BR Port and setting its attributes 19](#_Toc461043502)

[6.3.1 Creating 802.1BR UPSTREAM Port 19](#_Toc461043503)

[6.3.2 Creating 802.1BR CASCADING Port 20](#_Toc461043504)

[6.3.3 Creating 802.1BR ACCESS Port 20](#_Toc461043505)

[6.3.4 Deleting 802.1BR Port 20](#_Toc461043506)

[6.3.5 Setting 802.1BR Port ECID 20](#_Toc461043507)

[6.3.6 Setting 802.1BR Port PCP 21](#_Toc461043508)

[6.3.7 Setting 802.1BR Port DEI 21](#_Toc461043509)

[6.4 Setting Port Attributes 21](#_Toc461043510)

[6.4.1 Setting 802.1BR Port Discard Untagged frames 21](#_Toc461043511)

[6.4.2 Setting 802.1BR Port Discard Tagged frames 21](#_Toc461043512)

[6.5 802.1BR ECID Forwarding Entry Management 22](#_Toc461043513)

[6.5.1 Creating 802.1BR ECID Forwarding Entry 22](#_Toc461043514)

[6.5.2 Modifying the 802.1BR ECID Forwarding Entry 22](#_Toc461043515)

[6.5.3 Deleting 802.1BR ECID Forwarding Entry 23](#_Toc461043516)

[7 Summary of the Configurations 24](#_Toc461043517)

[7.1 Configurations at CB 24](#_Toc461043518)

[7.2 Configurations at PE 24](#_Toc461043519)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| Initial Version | Proposal for 802.1BR – Base Version | Ravikumar Sivasankar | 13 Jan 2016 |
| 0.1 | In the “Introduction” section including the diagram, corrected the wrong usage of the Source and Destination ECID fields. | Ravikumar Sivasankar | 20 Jan 2016 |
| 0.2 | * Added the attribute ‘SAI\_VLAN\_ATTR\_FLOODING\_ECID’ to Vlan api. * Allowed CB Extended ports to be added/removed to/from the Vlan * Updated the “Configuration Example” section accordingly | Ravikumar Sivasankar | 28 Jan 2016 |
| 0.3 | Added the missing attribute SAI\_DOT1BR\_PORT\_ATTR\_PORT in inc/saidot1brport.h file.  Updated the corresponding examples section. | Ravikumar Sivasankar | 20 Jun 2016 |
| 0.4 | Updated as per the Dot1BR Pipeline model | Ravikumar Sivasankar | 03 Aug 2016 |
| 0.5 | Removed the restriction of the port having to be a dot1br access, for setting the Port ECID, PCP and DEI | Ravikumar Sivasankar | 04 Aug 2016 |
| 0.6 | Added the relationship between Dot1br functionality and Tunnel model | Ravikumar Sivasankar | 03 Sep 2016 |
| 0.7 | * Added the support for LAG with Extended Ports. * Added xSTP on Extended Ports | Ravikumar Sivasankar | 07 Sep 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

The proposal addresses the IEEE 802.1BR functionality in SAI.

Definition and Acronyms

|  |  |
| --- | --- |
| CB | Controlling Bridge |
| ECID | E-Channel Id |
| PE | Port Extender |

# Overview

## 802.1BR Introduction



The Controlling Bridge (CB) discovers that the PEs – PE1 and PE2 - are reachable through the cascading ports CP1 and CP2 respectively. The CB will see the PE Access Ports as extended ports. It will assign ECID (E Channel Id) to the PE Access ports.

For the above topology, let the extended ports for PE Access Ports (AP1, AP2, AP3 and AP4) be EP1, EP2, EP3 and EP4 respectively.

The communication between the hosts consists of

* Upstream traffic - From source host to CB
* Downstream traffic - From CB to the destination host

Traffic flow from Host A to Host C is considered as an example for the following sections. The hosts A to D are considered to be in the same vlan, say Vlan X.

Upstream traffic

* PE will insert 802.1BR tag for all the traffic received on the Access Ports from the hosts. The ECID field of the 802.1BR tag will be set to the Access Port ECID. The Ingress ECID field will be set to 0.
* The traffic is sent on the Upstream Port towards the CB

Downstream traffic

CB Processing

* From the ETAG ECID field and the ingress port (CP1), CB will identify that the traffic is received on the extended port EP1.
* The ETAG header will be stripped
* It will learn Source Mac address (Mac-A) on EP1.
* The Destination Mac address (Mac-C) is looked up on the L2 Table.
* If the destination mac is known
  + The lookup result will yield EP3 as the egress port.
  + New ETAG is inserted with Ingress ECID field as 0 and ECID field as ECID-C
* If the destination mac is unknown
  + The packet will be flooded to all the hosts (which are in the same vlan). The ETAG will be inserted in the packet.
    - The Ingress ECID field will be set to the ECID field (ECID-A) of the incoming packet.
    - The ECID field will be set to the Multicast ECID assigned to the Vlan.
* The traffic (with ETAG header) is sent on the Cascading port CP2 towards the PE

PE Processing

* On receiving the traffic on the Upstream port, PE will perform lookup based on the ETAG.
* The ECID field in the ETAG header will be used to obtain the Egress port, which will be AP3 in this example.

### Forwarding Tables at CB

Extended Port Assignment Table

|  |  |  |
| --- | --- | --- |
| Cascading Port  (Key) | ECID  (Key) | Virtual/Extended Port  (Attribute) |
| CP1 | ECID-A | EP1 |
| CP1 | ECID-B | EP2 |
| CP2 | ECID-C | EP3 |
| CP2 | ECID-D | EP4 |

FDB Table

|  |  |  |
| --- | --- | --- |
| Vlan  (Key) | Mac Addr  (Key) | Egress Port  (Attribute) |
| X | MAC-A | EP1 |
| X | MAC-B | EP2 |
| X | MAC-C | EP3 |
| X | MAC-D | EP4 |

### Forwarding Table at PE1

|  |  |
| --- | --- |
| ECID  (Key) | Egress Port  (Attribute) |
| ECID-A | AP1 |
| ECID-B | AP2 |

### Forwarding Table at PE2

|  |  |
| --- | --- |
| ECID  (Key) | Egress Port  (Attribute) |
| ECID-C | AP2 |
| ECID-D | AP3 |

## Lag of Extended Ports

A LAG can be formed of Extended Ports. But all the members must be Extended Ports.



The diagram on the left above shows a CB connected to 2 PEs.

PE1 is connected to CB by Cascading Lag CP1.

PE2 is connected to CB by Cascading Lag CP2.

The PE ports (AP1, AP2, AP3, AP4, AP5) are instantiated as EP1, EP2, EP3, EP4, EP5 in CB respectively. The ECIDs are assigned to the PE Access Ports as follows:

ECID of APi = ECID-APi, where I = 1, 2, 3, 4, 5

Each Extended Port in CB is identified as follows:

EP1 = {Cascading Lag CP1, ECID-AP1}

EP2 = {Cascading Lag CP1, ECID-AP2}

EP3 = {Cascading Lag CP1, ECID-AP3}

EP4 = {Cascading Lag CP2, ECID-AP4}

EP5 = {Cascading Lag CP2, ECID-AP5}

There are 2 LAGs formed at the CB.

Lag1 is formed over the extended Ports EP1, EP2

Lag2 is formed over the Extended Ports EP3, EP4. Note that this Lag spans across PEs.

The Server A is connected to the CB-PE system through a Lag1.

The Server B is connected to the CP-PE system through a Lag2.

The right portion of the above diagram shows how it would appear to the external Servers. They would see as though they are connected to a single system.

When the CB transmits a frame to Server A, it choses Lag1 as the egress port. It will perform the following actions in the order shown below:

* Uses Lag hashing to pick one of the 2 members (EP1, EP2)
* It then determines that the chosen Lag member is reachable through the Cascading Lag, CP1.
* It then uses Lag hashing to pick one of the members of the Cascading Lag CP1.

## xSTP on Extended Ports



xSTP Protocols can run over Extended Ports. The above diagram illustrates how xSTP operates over Extended Ports.

The xSTP protocol will run only in the CB and not on the PE. In the above diagram the xSTP Bridge is considered to be a Root Bridge. As a result of this the xSTP protocol in CB places the Extended Ports EP2 and EP3 in Blocking state. It then informs the PE to place the Ports AP2 and AP3 in Blocking state.

Setting the xSTP states on the Ports in the PE ensures that the frames are dropped (in case of blocking state) on the PE, instead of traversing till the CB and then getting dropped.

# Dot1br v/s Tunnel model

Dot1br does not fit well into the Tunnel model, as a result of the following:

* **Dot1br header is just a tag**

Unlike the other Tunnel types, dot1br does not have an overlay and an underlay network. Dot1br header is just an extension. The data frame is not encapsulated in a dot1br header.

* **Extended Ports are merely an extension of local ports in CB**

The CB and its associated PE sub systems should be viewed as a single device with ports being remotely located in a remote device. All the Port attributes are applicable to the Extended ports. It is just that these Port attributes are applicable within the scope of the PE and not within the scope of the CB.

It is similar to a Chassis based system, with CB being the control card and the PEs being the Linecards

* **Asymmetric traffic flow.**

The traffic flow in the dot1br domain is asymmetric. The upstream flow (from PE ports to CB) can neither be treated as unicast or multicast. The downstream flow is either unicast or multicast based on the L2 Table lookup.

If we were to map the dot1br functionality to Tunnel model, then there should be support for Point to Multipoint Tunnels (single source, multiple receivers) for handling multicast traffic at CB.

# Specification

## Changes to saitypes.h

typedef enum \_sai\_object\_type\_t {

…

SAI\_OBJECT\_TYPE\_DOT1BR\_EXTENDED\_PORT = 30,

SAI\_OBJECT\_TYPE\_DOT1BR\_PORT = 31,

SAI\_OBJECT\_TYPE\_DOT1BR\_ECID\_FWD\_ENTRY = 32,

} sai\_object\_type\_t;

## Changes to sai.h

typedef enum \_sai\_api\_t

{

…

SAI\_API\_DOT1BR\_EXTENDED\_PORT = 26, /\*\*< sai\_dot1br\_extended\_port\_api\_t \*/

SAI\_API\_DOT1BR\_PORT = 27, /\*\*< sai\_dot1br\_port\_api\_t \*/

SAI\_API\_DOT1BR\_ECID\_FWD\_ENTRY = 28, /\*\*< sai\_dot1br\_ecid\_fwd\_entry\_api\_t \*/

} sai\_api\_t;

## Changes to saiport.h

typedef enum \_sai\_port\_attr\_t

{

/\*\* NOT Applicable to 802.1BR Extended Ports \*/

/\*\* READ-ONLY \*/

…

/\*\* READ-WRITE \*/

…

/\*\* Dropping of 802.1BR untagged frames on ingress [bool] (default to FALSE).

\* Applicable only to Physical ports. \*/

SAI\_PORT\_ATTR\_DOT1BR\_DROP\_UNTAGGED,

/\*\* Dropping of 802.1BR tagged frames on ingress [bool] (default to FALSE)

\* Applicable only to Physical ports. \*/

SAI\_PORT\_ATTR\_DOT1BR\_DROP\_TAGGED,

} sai\_port\_attr\_t;

## Changes to saifdb.h

/\*\*

\* @brief Attribute Id for fdb entry

\*/

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

{

…

/\*\* 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, SAI 802.1BR Extended port object id, etc. on. \*/

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

…

} sai\_fdb\_entry\_attr\_t;

## New File saidot1brport.h

/\*\*

\* Copyright (c) 2015 Dell 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.

\*

\*/

/\*\*

\* Module Name:

\*

\* saidot1brport.h

\*

\* Abstract:

\*

\* This module defines SAI API for IEEE 802.1BR Port attributes.

\*

\*/

#if !defined (\_\_SAIDOT1BRPORT\_H)

#define \_\_SAIDOT1BRPORT\_H

#include "saitypes.h"

#include "saistatus.h"

/\*\* \defgroup SAIDOT1BRPORT SAI - 802.1BR Port specific public APIs and datastructures.

\*

\* \{

\*/

/\*\*

\* @brief Attribute data for SAI\_DOT1BR\_PORT\_ATTR\_TYPE

\*/

typedef enum \_sai\_dot1br\_port\_type\_t

{

SAI\_DOT1BR\_PORT\_TYPE\_NONE,

SAI\_DOT1BR\_PORT\_TYPE\_UPSTREAM,

SAI\_DOT1BR\_PORT\_TYPE\_CASCADE,

SAI\_DOT1BR\_PORT\_TYPE\_ACCESS,

} sai\_dot1br\_port\_type\_t;

/\*\*

\* @brief SAI attributes for SAI\_OBJECT\_TYPE\_DOT1BR\_PORT \*/

typedef enum \_sai\_dot1br\_port\_attr\_t

{

/\*\* READ-WRITE \*/

   /\*\* The Port to which the 802.1BR Port is mapped to [sai\_object\_id\_t]  (MANDATORY\_ON\_CREATE|CREATE\_ONLY).

     \* Applicable only to Physical ports. \*/

   SAI\_DOT1BR\_PORT\_ATTR\_PORT,

/\*\* 802.1BR Port Type [sai\_dot1br\_port\_type\_t]

\* (MANDATORY\_ON\_CREATE|CREATE\_ONLY).

\* Applicable only to Physical ports. \*/

SAI\_DOT1BR\_PORT\_ATTR\_TYPE,

/\*\* 802.1BR Port default ECID [sai\_uint32\_t] (CREATE\_AND\_SET) (default to 0).

\* ECID to be added on receiving dot1br untagged frames.

\* Applicable only to Physical ports. \*/

SAI\_DOT1BR\_PORT\_ATTR\_ECID,

/\*\* 802.1BR Port default PCP [sai\_uint8\_t] (CREATE\_AND\_SET) (default to 0).

\* PCP to be added on receiving dot1br untagged frames.

\* Applicable only to Physical ports. \*/

SAI\_DOT1BR\_PORT\_ATTR\_PCP,

/\*\* 802.1BR Port default DEI [sai\_uint8\_t] (CREATE\_AND\_SET) (default to 0).

\* DEI to be added on receiving dot1br untagged frames.

\* Applicable only to Physical ports. \*/

SAI\_DOT1BR\_PORT\_ATTR\_DEI,

/\* -- \*/

/\* Custom range base value \*/

SAI\_DOT1BR\_PORT\_ATTR\_CUSTOM\_RANGE\_BASE = 0x10000000

} sai\_dot1br\_port\_attr\_t;

/\*\*

\* @brief Create a 802.1BR port.

\*

\* @param[out] dot1br\_port\_id

\* @param[in] attr\_count Number of attributes

\* @param[in] attr\_list Value of attributes

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_create\_dot1br\_port\_fn)(

\_Out\_ sai\_object\_id\_t \*dot1br\_port\_id,

\_In\_ uint32\_t attr\_count,

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

/\*\*

\* @brief Remove dot1br port.

\*

\* @param[in] dot1br\_port\_id Dot1BR Port object id.

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_dot1br\_port\_fn)(

\_In\_ sai\_object\_id\_t dot1br\_port\_id);

/\*\*

\* @brief Set the attribute of the Dot1BR Port.

\*

\* @param[in] dot1br\_port\_id Dot1BR Port object id.

\* @param[in] attr attribute value

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_dot1br\_port\_attribute\_fn)(

\_In\_ sai\_object\_id\_t dot1br\_port\_id,

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

/\*\*

\* @brief Get the attribute of Extended Port.

\*

\* @param[in] dot1br\_port\_id Dot1BR Port object id.

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

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

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_get\_dot1br\_port\_attribute\_fn)(

\_In\_ sai\_object\_id\_t dot1br\_port\_id,

\_In\_ uint32\_t attr\_count,

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

/\*\*

\* @brief SAI\_OBJECT\_TYPE\_DOT1BR\_PORT method table retrieved with sai\_api\_query()

\*/

typedef struct \_sai\_dot1br\_port\_api\_t {

sai\_create\_dot1br\_port\_fn create\_dot1br\_port;

sai\_remove\_dot1br\_port\_fn remove\_dot1br\_port;

sai\_set\_dot1br\_port\_attribute\_fn set\_dot1br\_port\_attribute;

sai\_get\_dot1br\_port\_attribute\_fn get\_dot1br\_port\_attribute;

} sai\_dot1br\_port\_api\_t;

/\*\*

\* \}

\*/

#endif // \_\_SAIDOT1BRPORT\_H

## New File saidot1brextport.h

/\*\*

\* Copyright (c) 2015 Dell 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.

\*

\*/

/\*\*

\* Module Name:

\*

\* saidot1brextport.h

\*

\* Abstract:

\*

\* This module defines SAI API for IEEE 802.1BR Extended Port functionality

\*

\*/

#if !defined (\_\_SAIDOT1BREXTPORT\_H)

#define \_\_SAIDOT1BREXTPORT\_H

#include "saitypes.h"

#include "saistatus.h"

/\*\* \defgroup SAIDOT1BREXTPORT SAI - 802.1BR Extended Port specific public APIs and datastructures.

\*

\* \{

\*/

/\*\*

\* @brief SAI attributes for SAI\_OBJECT\_TYPE\_DOT1BR\_EXTENDED\_PORT

\*/

typedef enum \_sai\_dot1br\_extended\_port\_attr\_t

{

/\*\* READ-WRITE \*/

/\*\* Cascading Port [sai\_object\_id\_t]

\* (MANDATORY\_ON\_CREATE|CREATE\_AND\_SET) \*/

SAI\_DOT1BR\_EXTENDED\_PORT\_ATTR\_CASCADING\_PORT,

/\*\* E-Channel Id (ECID) of the Extended Port [sai\_uint32\_t]

\* (MANDATORY\_ON\_CREATE|CREATE\_AND\_SET \*/

SAI\_DOT1BR\_EXTENDED\_PORT\_ATTR\_ECID,

/\* -- \*/

/\* Custom range base value \*/

SAI\_DOT1BR\_EXTENDED\_PORT\_ATTR\_CUSTOM\_RANGE\_BASE = 0x10000000

} sai\_dot1br\_extended\_port\_attr\_t;

/\*\*

\* @brief Create a 802.1BR extended port.

\*

\* @param[out] extended\_port\_id

\* @param[in] attr\_count Number of attributes

\* @param[in] attr\_list Value of attributes

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_create\_extended\_port\_fn)(

\_Out\_ sai\_object\_id\_t \*extended\_port\_id,

\_In\_ uint32\_t attr\_count,

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

/\*\*

\* @brief Remove extended port.

\*

\* @param[in] extended\_port\_id Extended Port object id.

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_extended\_port\_fn)(

\_In\_ sai\_object\_id\_t extended\_port\_id);

/\*\*

\* @brief Set the attribute of the Extended Port.

\*

\* @param[in] extended\_port\_id Extended Port object id.

\* @param[in] attr attribute value

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_extended\_port\_attribute\_fn)(

\_In\_ sai\_object\_id\_t extended\_port\_id,

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

/\*\*

\* @brief Get the attribute of Extended Port.

\*

\* @param[in] extended\_port\_id Extended Port object id.

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

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

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_get\_extended\_port\_attribute\_fn)(

\_In\_ sai\_object\_id\_t extended\_port\_id,

\_In\_ uint32\_t attr\_count,

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

/\*\*

\* @brief SAI\_OBJECT\_TYPE\_DOT1BR\_EXTENDED\_PORT method table retrieved with sai\_api\_query()

\*/

typedef struct \_sai\_dot1br\_extended\_port\_api\_t {

sai\_create\_extended\_port\_fn create\_extended\_port;

sai\_remove\_extended\_port\_fn remove\_extended\_port;

sai\_set\_extended\_port\_attribute\_fn set\_extended\_port\_attribute;

sai\_get\_extended\_port\_attribute\_fn get\_extended\_port\_attribute;

} sai\_dot1br\_extended\_port\_api\_t;

/\*\*

\* \}

\*/

#endif // \_\_SAIDOT1BREXTPORT\_H

## Changes to saivlan.h

/\*\*

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

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

\*/

typedef enum \_sai\_vlan\_attr\_t

{

/\*\* READ-ONLY \*/

…

/\*\* READ-WRITE \*/

…

/\*\*

\* The Multicast 802.1BR E-Channel Id (ECID), to be used in the flooding traffic

\* sent by the CB to Port Extenders (PE). [sai\_uint32\_t] (CREATE\_AND\_SET) (default to 0)

\*/

SAI\_VLAN\_ATTR\_FLOODING\_ECID,

…

} sai\_vlan\_attr\_t;

/\*\*

\* Routine Description:

\* @brief Add Port to VLAN

\*

\* Arguments:

\* @param[in] vlan\_id - VLAN id

\* @param[in] port\_count - number of ports

\* @param[in] port\_list - pointer to membership structures. The port list

\* can also include 802.1BR Extended Ports

\* (SAI\_OBJECT\_TYPE\_DOT1BR\_EXTENDED\_PORT)

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_add\_ports\_to\_vlan\_fn)(

\_In\_ sai\_vlan\_id\_t vlan\_id,

\_In\_ uint32\_t port\_count,

\_In\_ const sai\_vlan\_port\_t \*port\_list

);

/\*\*

\* Routine Description:

\* @brief Remove Port from VLAN

\*

\* Arguments:

\* @param[in] vlan\_id - VLAN id

\* @param[in] port\_count - number of ports

\* @param[in] port\_list - pointer to membership structures. The port list

\* can also include 802.1BR Extended Ports

\* (SAI\_OBJECT\_TYPE\_DOT1BR\_EXTENDED\_PORT)

\*

\* Return Values:

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_ports\_from\_vlan\_fn)(

\_In\_ sai\_vlan\_id\_t vlan\_id,

\_In\_ uint32\_t port\_count,

\_In\_ const sai\_vlan\_port\_t\* port\_list

);

## New file saidot1brecidfwd.h

/\*\*

\* Copyright (c) 2015 Dell 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.

\*

\*/

/\*\*

\* Module Name:

\*

\* saidot1brecidfwd.h

\*

\* Abstract:

\*

\* This module defines SAI API for IEEE 802.1BR ECID based forwarding functionality.

\*

\*/

#if !defined (\_\_SAIDOT1BRECIDFWD\_H)

#define \_\_SAIDOT1BRECID\_FWD\_H

#include "saitypes.h"

#include "saistatus.h"

/\*\* \defgroup SAIDOT1BRECIDFWDENTRY SAI - 802.1BR ECID based forwarding specific public APIs and datastructures.

\*

\* \{

\*/

/\*\*

\* @brief SAI attributes for SAI\_OBJECT\_TYPE\_DOT1BR\_ECID\_FWD\_ENTRY

\*/

typedef enum \_sai\_dot1br\_ecid\_fwd\_entry\_attr\_t

{

/\*\* READ-WRITE \*/

/\*\* 802.1BR ECID Forwarding entry ECID [sai\_uint32\_t] (MANDATORY\_ON\_CREATE)

\* The ECID for which the forwarding entry is to be created/set. Traffic

\* received, containing this ECID, will be

\* forwarded to the the port/portlist specified by SAI\_DOT1BR\_ECID\_FWD\_ENTRY\_ATTR\_PORT\_LIST.

I\*/

SAI\_DOT1BR\_ECID\_FWD\_ENTRY\_ECID,

/\*\* 802.1BR ECID Forwarding entry port list [sai\_object\_list\_t] (MANDATORY\_ON\_CREATE|CREATE\_AND\_SET)

\* The port id in the port list here can refer to a generic port object such as

\* SAI port object id, SAI LAG object id but not SAI DOT1BR Extended Port.

\* If the ECID associated with the dot1br Ecid Forwarding entry is an unicast ECID, then the port list

\* MUST contain only one port.

\* If the ECID associated with the dot1br Ecid Forwarding entry is a multicast ECID, then the port

\* list will overwrite the already present port list if any. \*/

SAI\_DOT1BR\_ECID\_FWD\_ENTRY\_ATTR\_PORT\_LIST,

/\* -- \*/

/\* Custom range base value \*/

SAI\_DOT1BR\_ECID\_FWD\_ENTRY\_ATTR\_CUSTOM\_RANGE\_BASE = 0x10000000

} sai\_dot1br\_ecid\_fwd\_entry\_attr\_t;

/\*\*

\* @brief Create a 802.1BR ECID Forwarding entry.

\*

\* @param[out] dot1br\_ecid\_fwd\_entry\_id Dot1br ECID Fwd entry Object Id

\* @param[in] attr\_count Number of attributes

\* @param[in] attr\_list Value of attributes

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_create\_dot1br\_ecid\_fwd\_entry\_fn)(

\_Out\_ sai\_object\_id\_t \*dot1br\_ecid\_fwd\_entry\_id,

\_In\_ uint32\_t attr\_count,

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

/\*\*

\* @brief Remove 802.1BR ECID Forwarding entry.

\*

\* @param[in] dot1br\_ecid\_fwd\_entry\_id Dot1br ECID Forwarding entry Object Id

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_dot1br\_ecid\_fwd\_entry\_fn)(

\_In\_ sai\_object\_id\_t dot1br\_ecid\_fwd\_entry\_id);

/\*\*

\* @brief Set the attribute of the 802.1BR ECID Fwd entry.

\*

\* @param[in] dot1br\_ecid\_fwd\_entry\_id Dot1br ECID Fwd entry Object Id

\* @param[in] attr attribute value

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_dot1br\_ecid\_fwd\_entry\_attribute\_fn)(

\_In\_ sai\_object\_id\_t dot1br\_ecid\_fwd\_entry\_id,

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

/\*\*

\* @brief Get the attribute of the 802.1BR ECID Fwd entry.

\*

\* @param[in] dot1br\_ecid\_fwd\_entry\_id Dot1br ECID Fwd entry Object Id

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

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

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_get\_dot1br\_ecid\_fwd\_entry\_attribute\_fn)(

\_In\_ sai\_object\_id\_t dot1br\_ecid\_fwd\_entry\_id,

\_In\_ uint32\_t attr\_count,

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

/\*\*

\* @brief SAI\_OBJECT\_TYPE\_DOT1BR\_ECID\_FWD\_ENTRY method table retrieved with sai\_api\_query().

\*/

typedef struct \_sai\_dot1br\_ecid\_fwd\_entry\_api\_t {

sai\_create\_dot1br\_ecid\_fwd\_entry\_fn create\_dot1br\_ecid\_fwd\_entry;

sai\_remove\_dot1br\_ecid\_fwd\_entry\_fn remove\_dot1br\_ecid\_fwd\_entry;

sai\_set\_dot1br\_ecid\_fwd\_entry\_attribute\_fn set\_dot1br\_ecid\_fwd\_entry\_attribute;

sai\_get\_dot1br\_ecid\_fwd\_entry\_attribute\_fn get\_dot1br\_ecid\_fwd\_entry\_attribute;

} sai\_dot1br\_ecid\_fwd\_entry\_api\_t;

/\*\*

\* \}

\*/

#endif // \_\_SAIDOT1BRECIDFWD\_H

# Configuration Example

## Creating and Deleting an Extended Port

The ports in the remote PE will be created as Extended Ports in CB. Each Extended Port is identified by the Cascading Port (through with the associated PE is reachable) and the ECID assigned to the Extended Port.

Creating an Extended Port

sai\_object\_id\_t extended\_port\_id;

sai\_object\_id\_t cascading\_port\_id;

sai\_uint32\_t ecid;

sai\_uint32\_t attr\_count = 2;

sai\_attribute\_t attr\_list [2];

attr\_list [0].id = SAI\_DOT1BR\_EXTENDED\_PORT\_ATTR\_CASCADING\_PORT;

attr\_list [0].value.object\_id = cascading\_port\_id;

attr\_list [1].id = SAI\_DOT1BR\_EXTENDED\_PORT\_ATTR\_ECID;

attr\_list [1].value.u32 = ecid;

sai\_create\_extended\_port\_fn (&extended\_port\_id, attr\_count, attr\_list);

Deleting an Extended Port

sai\_remove\_extended\_port\_fn (extended\_port\_id);

## Vlan configuration

### Adding extended ports to the Vlan

sai\_vlan\_id\_t vlan\_id;

sai\_object\_id\_t extended\_port\_id\_1;

sai\_object\_id\_t extended\_port\_id\_2;

sai\_object\_id\_t untagged\_port\_id;

sai\_object\_id\_t tagged\_port\_id;

sai\_uint32\_t port\_count = 4;

sai\_vlan\_port\_t port\_list [4];

port\_list [0].port\_id = untagged\_port\_id;

port\_list [0].tagging\_mode = SAI\_VLAN\_PORT\_UNTAGGED;

port\_list [1].port\_id = tagged\_port\_id;

port\_list [1].tagging\_mode = SAI\_VLAN\_PORT\_TAGGED;

port\_list [2].port\_id = extended\_port\_id\_1;

port\_list [2].tagging\_mode = SAI\_VLAN\_PORT\_TAGGED;

port\_list [3].port\_id = extended\_port\_id\_2;

port\_list [3].tagging\_mode = SAI\_VLAN\_PORT\_TAGGED;

sai\_add\_ports\_to\_vlan\_fn (vlan\_id, port\_count, &port\_list [0]);

### Assigning Flooding ECID to the Vlan

sai\_vlan\_id\_t vlan\_id;

sai\_uint32\_t flooding\_ecid;

sai\_attribute\_t attr;

attr.id = SAI\_VLAN\_ATTR\_FLOODING\_ECID;

attr.value.u32 = flooding\_ecid;

sai\_set\_vlan\_attribute\_fn (vlan\_id, &attr);

### Removing extended ports from the Vlan

sai\_vlan\_id\_t vlan\_id;

sai\_object\_id\_t extended\_port\_id;

sai\_object\_id\_t untagged\_port\_id;

sai\_uint32\_t port\_count = 2;

sai\_vlan\_port\_t port\_list [2];

port\_list [0].port\_id = untagged\_port\_id;

port\_list [0].tagging\_mode = SAI\_VLAN\_PORT\_UNTAGGED;

port\_list [2].port\_id = extended\_port\_id;

port\_list [2].tagging\_mode = SAI\_VLAN\_PORT\_TAGGED;

sai\_remove\_ports\_from\_vlan\_fn (vlan\_id, port\_count, &port\_list [0]);

## Creating/Deleting 802.1BR Port and setting its attributes

### Creating 802.1BR UPSTREAM Port

sai\_object\_id\_t dot1br\_port\_id;

sai\_object\_id\_t port\_id;

sai\_uint32\_t ecid;

sai\_uint32\_t attr\_count = 2;

sai\_attribute\_t attr\_list [2];

attr\_list [0].id = SAI\_DOT1BR\_PORT\_ATTR\_PORT;

attr\_list [0].value.oid = port\_id;

attr\_list [1].id = SAI\_DOT1BR\_PORT\_ATTR\_TYPE;

attr\_list [1].value.s32 = SAI\_DOT1BR\_PORT\_TYPE\_UPSTREAM;

sai\_create\_dot1br\_port\_fn (&dot1br\_port\_id, attr\_count, attr\_list);

### Creating 802.1BR CASCADING Port

sai\_object\_id\_t dot1br\_port\_id;

sai\_object\_id\_t port\_id;

sai\_uint32\_t ecid;

sai\_uint32\_t attr\_count = 2;

sai\_attribute\_t attr\_list [2];

attr\_list [0].id = SAI\_DOT1BR\_PORT\_ATTR\_PORT;

attr\_list [0].value.oid = port\_id;

attr\_list [1].id = SAI\_DOT1BR\_PORT\_ATTR\_TYPE;

attr\_list [1].value.s32 = SAI\_DOT1BR\_PORT\_TYPE\_CASCADING;

sai\_create\_dot1br\_port\_fn (&dot1br\_port\_id, attr\_count, attr\_list);

### Creating 802.1BR ACCESS Port

sai\_object\_id\_t dot1br\_port\_id;

sai\_object\_id\_t port\_id;

sai\_uint32\_t ecid;

sai\_uint32\_t attr\_count = 3;

sai\_attribute\_t attr\_list [3];

attr\_list [0].id = SAI\_DOT1BR\_PORT\_ATTR\_PORT;

attr\_list [0].value.oid = port\_id;

attr\_list [1].id = SAI\_DOT1BR\_PORT\_ATTR\_TYPE;

attr\_list [1].value.s32 = SAI\_DOT1BR\_PORT\_TYPE\_ACCESS;

attr\_list [2].id = SAI\_DOT1BR\_PORT\_ATTR\_ECID;

attr\_list [2].value.u32 = ecid;

sai\_create\_dot1br\_port\_fn (&dot1br\_port\_id, attr\_count, attr\_list);

### Deleting 802.1BR Port

sai\_object\_id\_t dot1br\_port\_id;

sai\_remove\_dot1br\_port\_fn (dot1br\_port\_id);

### Setting 802.1BR Port ECID

The device should insert the Port ECID for 802.1BR untagged frames received on the Access Ports.

sai\_object\_id\_t dot1br\_port\_id;

sai\_uint32\_t ecid;

sai\_attribute\_t attr;

attr.id = SAI\_DOT1BR\_PORT\_ATTR\_ECID;

attr.value.u32 = ecid;

sai\_set\_dot1br\_port\_attribute\_fn (dot1br\_port\_id, &attr);

### Setting 802.1BR Port PCP

The device should insert the Port PCP for 802.1BR untagged frames received on the Access Ports.

sai\_object\_id\_t dot1br\_port\_id;

sai\_uint8\_t pcp;

sai\_attribute\_t attr;

attr.id = SAI\_DOT1BR\_PORT\_ATTR\_PCP;

attr.value.u8 = pcp;

sai\_set\_dot1br\_port\_attribute\_fn (dot1br\_port\_id, &attr);

### Setting 802.1BR Port DEI

The device should insert the Port DEI for 802.1BR untagged frames received on the Access Ports.

sai\_object\_id\_t dot1br\_port\_id;

sai\_uint8\_t dei;

sai\_attribute\_t attr;

attr.id = SAI\_DOT1BR\_PORT\_ATTR\_DEI;

attr.value.u8 = dei;

sai\_set\_dot1br\_port\_attribute\_fn (dot1br\_port\_id, &attr);

## Setting Port Attributes

### Setting 802.1BR Port Discard Untagged frames

This attribute specifies if 802.1BR untagged frames should be allowed or dropped on ingress.

sai\_object\_id\_t port\_id;

sai\_attribute\_t port\_attr\_set;

port\_attr\_set.id = SAI\_PORT\_ATTR\_DOT1BR\_DROP\_UNTAGGED;

port\_attr\_set.value.bool = TRUE; /\* 802.1BR untagged frames will always be dropped on ingress \*/

sai\_set\_port\_attribute\_fn (port\_id, &port\_attr\_set);

### Setting 802.1BR Port Discard Tagged frames

This attribute specifies if 802.1BR tagged frames should be allowed or dropped on ingress.

sai\_object\_id\_t port\_id;

sai\_attribute\_t port\_attr\_set;

port\_attr\_set.id = SAI\_PORT\_ATTR\_DOT1BR\_DROP\_TAGGED;

port\_attr\_set.value.bool = TRUE; /\* 802.1BR agged frames will always be dropped on ingress \*/

sai\_set\_port\_attribute\_fn (port\_id, &port\_attr\_set);

## 802.1BR ECID Forwarding Entry Management

In PE, the downstream traffic (traffic from the CB/Transit PE received on the Upstream Port) will be forwarded based on the ECID present in the traffic. This is achieved using the 802.1BR ECID Forwarding table.

### Creating 802.1BR ECID Forwarding Entry

Creating Unicast 802.1BR ECID Forwarding Entry

sai\_object\_id\_t ecid\_fwd\_entry\_id;

sai\_object\_id\_t egress\_port\_id;

sai\_uint32\_t unicast\_ecid;

sai\_uint32\_t attr\_count = 2;

sai\_attribute\_t attr\_list [2];

attr\_list [0].id = SAI\_DOT1BR\_ECID\_FWD\_ENTRY\_ECID;

attr\_list [0].value.u32 = unicast\_ecid;

attr\_list [1].id = SAI\_DOT1BR\_ECID\_FWD\_ENTRY\_ATTR\_PORT\_LIST;

attr\_list [1].value.objlist.count = 1;

attr\_list [1].value.objlist.list = &egress\_port\_id;

sai\_create\_dot1br\_ecid\_fwd\_entry\_fn (&ecid\_fwd\_entry\_id, attr\_count, attr\_list);

Creating Multicast 802.1BR ECID Forwarding Entry

sai\_object\_id\_t ecid\_fwd\_entry\_id;

sai\_object\_id\_t egress\_port\_id\_1;

sai\_object\_id\_t egress\_port\_id\_2;

sai\_uint32\_t multicast\_ecid;

sai\_uint32\_t attr\_count = 2;

sai\_attribute\_t attr\_list [2];

attr\_list [0].id = SAI\_DOT1BR\_ECID\_FWD\_ENTRY\_ECID;

attr\_list [0].value.u32 = multicast\_ecid;

attr\_list [1].value.objlist.list

= (sai\_object\_id\_t \*) calloc (2, sizeof (sai\_object\_id\_t));

attr\_list [1].id = SAI\_DOT1BR\_ECID\_FWD\_ENTRY\_ATTR\_PORT\_LIST;

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

attr\_list [1].value.objlist.list [0] = egress\_port\_id\_1;

attr\_list [1].value.objlist.list [1] = egress\_port\_id\_2;

sai\_create\_dot1br\_ecid\_fwd\_entry\_fn (&ecid\_fwd\_entry\_id, attr\_count, attr\_list);

### Modifying the 802.1BR ECID Forwarding Entry

Modifying Unicast 802.1BR ECID Forwarding Entry

sai\_object\_id\_t ecid\_fwd\_entry\_id;

sai\_object\_id\_t egress\_port\_id\_new;

sai\_uint32\_t attr\_count = 1;

sai\_attribute\_t attr\_list [1];

attr\_list [0].id = SAI\_DOT1BR\_ECID\_FWD\_ENTRY\_ATTR\_PORT\_LIST;

attr\_list [0].value.objlist.count = 1;

attr\_list [0].value.objlist.list = &egress\_port\_id\_new;

sai\_set\_dot1br\_ecid\_fwd\_entry\_fn (ecid\_fwd\_entry\_id, attr\_count, attr\_list);

Modifying Multicast 802.1BR ECID Forwarding Entry

sai\_object\_id\_t ecid\_fwd\_entry\_id;

sai\_object\_id\_t egress\_port\_id\_3;

sai\_object\_id\_t egress\_port\_id\_4;

sai\_object\_id\_t egress\_port\_id\_5;

sai\_uint32\_t attr\_count = 1;

sai\_attribute\_t attr\_list [1];

attr\_list [0].value.objlist.list

= (sai\_object\_id\_t \*) calloc (3, sizeof (sai\_object\_id\_t));

attr\_list [0].id = SAI\_DOT1BR\_ECID\_FWD\_ENTRY\_ATTR\_PORT\_LIST;

attr\_list [0].value.objlist.count = 3;

attr\_list [0].value.objlist.list [0] = egress\_port\_id\_3;

attr\_list [0].value.objlist.list [1] = egress\_port\_id\_4;

attr\_list [0].value.objlist.list [2] = egress\_port\_id\_5;

sai\_set\_dot1br\_ecid\_fwd\_entry\_fn (ecid\_fwd\_entry\_id, attr\_count, attr\_list);

### Deleting 802.1BR ECID Forwarding Entry

sai\_object\_id\_t ecid\_fwd\_entry\_id;

sai\_remove\_dot1br\_ecid\_fwd\_entry\_fn (ecid\_fwd\_entry\_id);

# Summary of the Configurations

## Configurations at CB

The Port connected to the PE is called as Cascading Port.

* Create the 802.1BR Cascading Port. See Section 6.3.2 for configuration example.
* Create the Extended Port using the Cascading Port and the ECID assigned to the remote PE Port. See Section 6.1 for configuration example.
* Assign Flooding ECID to the Vlans. See Section 6.2.2 for configuration example.
* Assign CB Extended Ports to the Vlans. See Section 6.2.1 for configuration example.

## Configurations at PE

The Port connected to the CB is called as Upstream Port.

The Ports connected to the Hosts are called Access Ports.

A PE can be connected to a downstream PE (This is not shown in the topology diagram in Section 2). These Ports are called Cascading Ports.

* Create 802.1BR Upstream Port. See Section 6.3.1 for configuration example.
* Create 802.1BR Cascading Port(s). See Section 6.3.2 for configuration example.
* Create 802.1BR Access Port(s). See Section 6.3.3 for configuration example.
* Set the Host Port PCP to the value that is to be inserted in the ETAG header of the received frames. See Section 6.3.6 for configuration example.
* Set the Host Port DEI to the value that is to be inserted in the ETAG header of the received frames. See Section 6.3.7 for configuration example.
* Install/Remove 802.1BR entries in the ECID forwarding table. See Sections 6.5.1, 6.5.2 and 6.5.3.