

|  |
| --- |
|  |

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](#_Toc441763151)

[1 Overview 1](#_Toc441763152)

[2 Introduction 2](#_Toc441763153)

[2.1 Forwarding Tables at CB 3](#_Toc441763154)

[2.2 Forwarding Table at PE1 4](#_Toc441763155)

[2.3 Forwarding Table at PE2 4](#_Toc441763156)

[3 SAI Pipeline Model 5](#_Toc441763157)

[3.1 CB Pipeline Model 5](#_Toc441763158)

[3.2 PE Pipeline Model 7](#_Toc441763159)

[4 Specification 8](#_Toc441763160)

[4.1 Changes to saitypes.h 8](#_Toc441763161)

[4.2 Changes to sai.h 8](#_Toc441763162)

[4.3 Changes to saiswitch.h 8](#_Toc441763163)

[4.4 Changes to saiport.h 9](#_Toc441763164)

[4.5 Changes to saifdb.h 9](#_Toc441763165)

[4.6 New File saidot1brport.h 9](#_Toc441763166)

[4.7 New File saidot1brcbextport.h 12](#_Toc441763167)

[4.8 Changes to saivlan.h 14](#_Toc441763168)

[4.9 New file saidot1brfdb.h 15](#_Toc441763169)

[5 Configuration Example 18](#_Toc441763170)

[5.1 Creating and Deleting an CB Extended Port (CB Only) 18](#_Toc441763171)

[5.2 Vlan configuration 18](#_Toc441763172)

[5.2.1 Adding CB extended ports to the Vlan 18](#_Toc441763173)

[5.2.2 Assigning Flooding ECID to the Vlan 19](#_Toc441763174)

[5.2.3 Removing CB extended ports from the Vlan 19](#_Toc441763175)

[5.3 Creating/Deleting 802.1BR Port and setting its attributes 19](#_Toc441763176)

[5.3.1 Creating 802.1BR UPSTREAM Port (PE Only) 19](#_Toc441763177)

[5.3.2 Creating 802.1BR CASCADING Port 20](#_Toc441763178)

[5.3.3 Creating 802.1BR ACCESS Port 20](#_Toc441763179)

[5.3.4 Deleting 802.1BR Port 20](#_Toc441763180)

[5.3.5 Setting 802.1BR Port ECID 20](#_Toc441763181)

[5.3.6 Setting 802.1BR Port PCP 20](#_Toc441763182)

[5.3.7 Setting 802.1BR Port DEI 21](#_Toc441763183)

[5.4 Setting Port Attributes 21](#_Toc441763184)

[5.4.1 Setting 802.1BR Port Discard Untagged frames 21](#_Toc441763185)

[5.4.2 Setting 802.1BR Port Discard Tagged frames 21](#_Toc441763186)

[5.5 802.1BR FDB Entry Management (PE Only) 21](#_Toc441763187)

[5.5.1 Creating 802.1BR FDB Entry 21](#_Toc441763188)

[5.5.2 Modifying the 802.1BR FDB Entry 22](#_Toc441763189)

[5.5.3 Deleting 802.1BR FDB Entry 23](#_Toc441763190)

[6 Summary of the Configurations 24](#_Toc441763191)

[6.1 Configurations at CB 24](#_Toc441763192)

[6.2 Configurations at PE 24](#_Toc441763193)

# 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 |
|  |  |  |  |
|  |  |  |  |

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 |

# 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 |

# SAI Pipeline Model

## CB Pipeline Model



NOTE:

The block diagram depicting the proposed SAI pipeline in the above diagram, may not be exactly the same as the one that was discussed during the OCP workshop. The one that was discussed during the workshop was projected on the screen and not checked-in in github repository. The diagram shown in this document is based on the discussion with the folks who attended the workshop.

In the above diagram, the OCP proposed Pipeline model is shown in solid block and the 802.1BR blocks are shown in dotted blocks.

The 802.1BR blocks perform the following functionalities:

On Ingress

1. On frame reception, if the frame is received as ETAGed packet on the Cascading Port (In Port), the Extended port is determined. Extended Port determination is a function of In Port and the Source ECID present in the ETAG header.
2. In Port is now set to the Extended Port determined at step 1.
3. ETAG header is stripped.

On Egress

1. If the Egress Port is an Extended Port, then insert ETAG header
   1. If the transmitted Frame is an Unicast Frame, then set the Ingress ECID field to 0, else
   2. If the transmitted Frame is a Multicast Frame, then set the Ingress ECID field to the Ingress ECID field of the ingress Frame.

Rest of the processing is based on the proposed Pipeline model.

## PE Pipeline Model



# Specification

## Changes to saitypes.h

typedef enum \_sai\_object\_type\_t {

…

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

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

SAI\_OBJECT\_TYPE\_DOT1BR\_FDB\_ENTRY = 32,

} sai\_object\_type\_t;

## Changes to sai.h

typedef enum \_sai\_api\_t

{

…

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

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

SAI\_API\_DOT1BR\_FDB\_ENTRY = 28, /\*\*< sai\_dot1br\_fdb\_entry\_api\_t \*/

} sai\_api\_t;

## Changes to saiswitch.h

/\*\*

\* @brief Attribute data for SAI\_SWITCH\_ATTR\_DOT1BR\_NODE\_TYPE

\*/

typedef enum \_sai\_switch\_dot1br\_node\_type\_t

{

/\*\* Controlling Bridge Node \*/

SAI\_SWITCH\_DOT1BR\_NODE\_TYPE\_CB,

/\*\* Port Extender Node \*/

SAI\_SWITCH\_DOT1BR\_NODE\_TYPE\_PE,

} sai\_switch\_dot1br\_node\_type;

typedef enum \_sai\_switch\_attr\_t

{

/\*\* READ-ONLY \*/

…

/\*\* switch's 802.1BR capability [bool] \*/

SAI\_SWITCH\_ATTR\_DOT1BR\_CAPABLE,

/\*\* 802.1BR Node type [sai\_switch\_dot1br\_node\_type] \*/

SAI\_SWITCH\_ATTR\_DOT1BR\_NODE\_TYPE,

/\*\* READ-WRITE \*/

…

} sai\_switch\_attr\_t;

NOTE:

In *sai\_switch\_attr\_t* new READ-ONLY attribute is added as the last read-only attribute. This will increment the enum values of all the READ-WRITE attributes.

## 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 CB 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] (MANDATORY\_ON\_CREATE|CREATE\_AND\_SET).

\* This attribute is valid only if the attribute SAI\_DOT1BR\_PORT\_ATTR\_TYPE

\* is set to SAI\_DOT1BR\_PORT\_TYPE\_ACCESS.

\* 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).

\* This attribute is valid only if the attribute SAI\_DOT1BR\_PORT\_ATTR\_TYPE

\* is set to SAI\_DOT1BR\_PORT\_TYPE\_ACCESS.

\* 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).

\* This attribute is valid only if the attribute SAI\_DOT1BR\_PORT\_ATTR\_TYPE

\* is set to SAI\_DOT1BR\_PORT\_TYPE\_ACCESS.

\* 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 saidot1brcbextport.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:

\*

\* saidot1brcbextport.h

\*

\* Abstract:

\*

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

\*

\*/

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

#define \_\_SAIDOT1BRPORT\_H

#include "saitypes.h"

#include "saistatus.h"

/\*\* \defgroup SAIDOT1BRCBEXTPORT SAI - 802.1BR Extension Port specific public APIs and datastructures in CB.

\*

\* \{

\*/

/\*\*

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

\*/

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

{

/\*\* READ-WRITE \*/

/\*\* Cascading Port in the Controlling Bridge [sai\_object\_id\_t]

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

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

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

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

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

/\* -- \*/

/\* Custom range base value \*/

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

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

/\*\*

\* @brief Create a 802.1BR extended port. This API is applicable only to Controlling Bridge.

\*

\* @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\_cb\_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\_cb\_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\_cb\_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\_cb\_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\_CB\_EXTENDED\_PORT method table retrieved with sai\_api\_query()

\*/

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

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

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

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

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

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

/\*\*

\* \}

\*/

#endif // \_\_SAIDOT1BRCBEXTPORT\_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 CB Extended Ports

\* (SAI\_OBJECT\_TYPE\_DOT1BR\_CB\_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 CB Extended Ports

\* (SAI\_OBJECT\_TYPE\_DOT1BR\_CB\_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 saidot1brfdb.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:

\*

\* saidot1brfdb.h

\*

\* Abstract:

\*

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

\* applicable only to Port Extenders (PE).

\*

\*/

#if !defined (\_\_SAIDOT1BRFDB\_H)

#define \_\_SAIDOT1BRFDB\_H

#include "saitypes.h"

#include "saistatus.h"

/\*\* \defgroup SAIDOT1BRFDBENTRY SAI - 802.1BR ECID based forwarding specific public APIs and datastructures in PE.

\*

\* \{

\*/

/\*\*

\* @brief SAI attributes for SAI\_OBJECT\_TYPE\_DOT1BR\_FDB\_ENTRY

\*/

typedef enum \_sai\_dot1br\_fdb\_entry\_attr\_t

{

/\*\* READ-WRITE \*/

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

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

\* received on the Upstream Port in PE, containing this ECID, will be

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

SAI\_DOT1BR\_FDB\_ENTRY\_ECID,

/\*\* 802.1BR FDB 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 CB Extended Port.

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

\* MUST contain only one port.

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

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

SAI\_DOT1BR\_FDB\_ENTRY\_ATTR\_PORT\_LIST,

/\* -- \*/

/\* Custom range base value \*/

SAI\_DOT1BR\_FDB\_ENTRY\_ATTR\_CUSTOM\_RANGE\_BASE = 0x10000000

} sai\_dot1br\_fdb\_entry\_attr\_t;

/\*\*

\* @brief Create a 802.1BR FDB entry.

\*

\* @param[out] dot1br\_fdb\_entry\_id Dot1br FDB 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\_fdb\_entry\_fn)(

\_Out\_ sai\_object\_id\_t \*dot1br\_fdb\_entry\_id,

\_In\_ uint32\_t attr\_count,

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

/\*\*

\* @brief Remove 802.1BR FDB entry.

\*

\* @param[in] dot1br\_fdb\_entry\_id Dot1br FDB entry Object Id

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

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

\_In\_ sai\_object\_id\_t dot1br\_fdb\_entry\_id);

/\*\*

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

\*

\* @param[in] dot1br\_fdb\_entry\_id Dot1br FDB 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\_fdb\_entry\_attribute\_fn)(

\_In\_ sai\_object\_id\_t dot1br\_fdb\_entry\_id,

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

/\*\*

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

\*

\* @param[in] dot1br\_fdb\_entry\_id Dot1br FDB 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\_fdb\_entry\_attribute\_fn)(

\_In\_ sai\_object\_id\_t dot1br\_fdb\_entry\_id,

\_In\_ uint32\_t attr\_count,

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

/\*\*

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

\* This API is applicable only to PE.

\*/

typedef struct \_sai\_dot1br\_fdb\_entry\_api\_t {

sai\_create\_dot1br\_fdb\_entry\_fn create\_dot1br\_fdb\_entry;

sai\_remove\_dot1br\_fdb\_entry\_fn remove\_dot1br\_fdb\_entry;

sai\_set\_dot1br\_fdb\_entry\_attribute\_fn set\_dot1br\_fdb\_entry\_attribute;

sai\_get\_dot1br\_fdb\_entry\_attribute\_fn get\_dot1br\_fdb\_entry\_attribute;

} sai\_dot1br\_fdb\_entry\_api\_t;

/\*\*

\* \}

\*/

#endif // \_\_SAIDOT1BRFDB\_H

# Configuration Example

## Creating and Deleting an CB Extended Port (CB Only)

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 cb\_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\_CB\_EXTENDED\_PORT\_ATTR\_CASCADING\_PORT;

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

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

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

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

Deleting an Extended Port

sai\_remove\_cb\_extended\_port\_fn (cb\_extended\_port\_id);

## Vlan configuration

### Adding CB extended ports to the Vlan

sai\_vlan\_id\_t vlan\_id;

sai\_object\_id\_t cb\_extended\_port\_id\_1;

sai\_object\_id\_t cb\_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 = cb\_extended\_port\_id\_1;

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

port\_list [3].port\_id = cb\_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 CB extended ports from the Vlan

sai\_vlan\_id\_t vlan\_id;

sai\_object\_id\_t cb\_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 = cb\_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 (PE Only)

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

Node 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

Node 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

Node 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 FDB Entry Management (PE Only)

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 FDB table.

### Creating 802.1BR FDB Entry

Creating Unicast 802.1BR FDB Entry

sai\_object\_id\_t fdb\_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\_FDB\_ENTRY\_ECID;

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

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

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

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

sai\_create\_dot1br\_fdb\_entry\_fn (&fdb\_entry\_id, attr\_count, attr\_list);

Creating Multicast 802.1BR FDB Entry

sai\_object\_id\_t fdb\_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\_FDB\_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\_FDB\_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\_fdb\_entry\_fn (&fdb\_entry\_id, attr\_count, attr\_list);

### Modifying the 802.1BR FDB Entry

Modifying Unicast 802.1BR FDB Entry

sai\_object\_id\_t fdb\_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\_FDB\_ENTRY\_ATTR\_PORT\_LIST;

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

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

sai\_set\_dot1br\_fdb\_entry\_fn (fdb\_entry\_id, attr\_count, attr\_list);

Modifying Multicast 802.1BR FDB Entry

sai\_object\_id\_t fdb\_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\_FDB\_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\_fdb\_entry\_fn (fdb\_entry\_id, attr\_count, attr\_list);

### Deleting 802.1BR FDB Entry

sai\_object\_id\_t fdb\_entry\_id;

sai\_remove\_dot1br\_fdb\_entry\_fn (fdb\_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 5.3.2 for configuration example.
* Create the CB Extended Port using the Cascading Port and the ECID assigned to the remote PE Port. See Section 5.1 for configuration example.
* Assign Flooding ECID to the Vlans. See Section 5.2.2 for configuration example.
* Assign CB Extended Ports to the Vlans. See Section 5.2.1 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 5.3.1 for configuration example.
* Create 802.1BR Cascading Port(s). See Section 5.3.2 for configuration example.
* Create 802.1BR Access Port(s). See Section 5.3.3 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 5.3.6 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 5.3.7 for configuration example.
* Install/Remove 802.1BR entries in the ECID forwarding table. See Sections 5.5.1, 5.5.2 and 5.5.3.