ZebOS-XP®
Network Platform

Version 1.4
Extended Performance

Data Center Bridging
Developer Guide

December 2015

IP Infusion Inc.
3965 Freedom Circle, Suite 200
Santa Clara, CA 95054
+1 408-400-1900
http://www.ipinfusion.com/

For support, questions, or comments via E-mail, contact:
support@ipinfusion.com

Contents

Preface

This guide describes the ZebOS-XP application programming interface (API) for Data Center Bridging (DCB).

Audience

This guide is intended for developers who write code to customize and extend DCB.

Conventions

Table P-1 shows the conventions used in this guide.

Table P-1: Conventions

Convention	Description
Italics	Emphasized terms; titles of books
Note:	Special instructions, suggestions, or warnings
`monospaced type`	Code elements such as commands, functions, parameters, files, and directories

Contents

This guide contains these chapters:

- Chapter 1, *Data Center Bridging*
- Chapter 2, *Data Structures*
- Chapter 3, *DCB-ETS Functions*
- Chapter 4, *DCB-PFC Functions*
- Chapter 5, *DCB-QCN Functions*

Related Documents

The following guides are related to this document:

- *Data Center Bridging Command Reference*
- *Data Center Bridging Configuration Guide*
- *Installation Guide*
- *Network Services Module Developer Guide*
- *Network Services Module Command Reference*

- *Architecture Guide*

Note: All ZebOS-XP technical manuals are available to licensed customers at http://www.ipinfusion.com/support/document_list.

Support

For support-related questions, contact support@ipinfusion.com.

Comments

If you have comments, or need to report a problem with the content, contact techpubs@ipinfusion.com.

CHAPTER 1 Data Center Bridging

Data Center Bridging (DCB) is a collection of standards-based extensions for Ethernet protocols. It provides a lossless transport layer to allow the convergence of LANs and SANs into a single unified fabric. DCB is a flexible framework that defines the capabilities required for switches and end-points to be part of a data center environment. Data Center Bridging protocols can carry Fibre Channel, TCP/IP, and IPC traffic over a single, converged 10 Gigabit Ethernet network.

Data center networks and backplane fabrics employ applications that depend on the delivery of data packets with a lower latency and lower probability of packet loss than is typical of IEEE 802 VLAN bridged networks. DCB supports the use of a single bridged local area network for these applications, as well as traditional LAN applications.

Data Center Bridging includes the following capabilities:

- Priority-based flow control (IEEE 802.1Qbb) - see Priority-based Flow Control (PFC) on page 10
- Enhanced transmission selection (IEEE 802.1Qaz) - see Enhanced Transmission Selection (ETS) on page 10
- Extensions to the Link Layer Discovery Protocol standard (IEEE 802.1AB) that enable Data Center Bridging Capability Exchange Protocol (DCBX) (IEEE 802.1Qaz) - see Data Center Bridging eXchange (DCBX) Protocol on page 10
- Quantized congestion notification (IEEE 802.1Qau) - see Quantized Congestion Notification (QCN) Protocol on page 10

Features

The following are some of the features of Data Center Bridging:

- Enhanced performance for Ethernet networks
- Lossless fabric for traffic on converged networks
- Virtual links by traffic class that includes pause-per-class technology, congestion management and event notification.
- Improved performance, including stability, enhanced throughput, and robustness
- Supports 802.1Qbb (PFC)
- Supports 802.1Qaz (ETS)

Data Centers

A Data Center is the core element of a network topology. An enterprise depends on it to run business operations, service providers depend on it to deliver network services, and content providers depend on it to distribute content. Thus, a data center must deliver reliability, availability, and high performance. The most widely deployed networking technology is Ethernet. It meets the current demands required by most businesses. However, Ethernet must also help data centers improve cost effectiveness and meet the demands of the next generation applications and services.

The Data Center Bridging standard can improve networks by implementing the same consolidated benefits that storage and servers have experienced for years; that is, higher utilization rates, simpler management, and lower total cost of ownership. DCB is reliable, provides predictable performance, and segregates and prioritize traffic. Typically, system administrators will implement a standard Ethernet network, a Data Center Bridging environment, or a combination of both.

System Architecture

The following section describes the system architecture of DCB:

Priority-based Flow Control (PFC)

Priority-based Flow Control (PFC) standard specifies protocols, procedures and managed objects that enable flow control per traffic class on IEEE 802 full-duplex links. Priority-based flow control helps eliminate frame loss due to congestion by operating on individual priorities. Along with other Data Center Bridging technologies, PFC helps the flow of higher-layer protocols, which are highly loss sensitive, while not affecting traditional LAN protocols that utilize other priorities.

Data Center Bridging networks (bridges and end-nodes) include limited bandwidth-delay product and limited hop-count. VLAN tag priority values identify traffic classes. Priority-based flow control helps eliminate frame loss due to congestion by a mechanism similar to the IEEE 802.3x PAUSE, but operating on individual priorities.

PFC uses a pause mechanism that allows a receiving device to signal a “pause” to the directly connected sending device prior to buffer overflow and/or packet loss. In the past, Ethernet had a tool to signal a pause to directly connected devices (802.3x pause); however, this was done at the link level. This would allow traffic on the link to be paused, but not a selected traffic type.

PFC does not pause an entire link, since pausing a link carrying various I/O types is not ideal, especially for traffic such as IP Telephony and streaming video. Instead, PFC sends a pause signal for a single Class of Service (CoS) that is part of an 802.1Q Ethernet header and does not pause an entire link. This allows up to eight (8) classes to be defined and paused independent of one another.

Enhanced Transmission Selection (ETS)

The Enhanced Transmission Selection (ETS) standard supports the allocation of bandwidth amongst traffic classes. When the available load in a traffic class does not use its allocated bandwidth, ETS allows other traffic classes to use the available bandwidth.

A network can prioritize traffic to offer different service characteristics to different traffic classes. However, a user may want to share bandwidth among the priorities that are carrying bursty, high-offered loads, instead of servicing them with a strict priority. This lets traffic set at a specific priority level that does not use its set allocation, to allow other priorities to be able to use the unused bandwidth.

For example, IEEE P802.1Qau specifies congestion management. Congestion managed traffic classes can share a network with traditional best-effort LAN classes. In addition, enhanced transmission selection provides uniform management for the sharing of bandwidth between congestion managed classes and traditional classes on a single bridged network. Priorities using enhanced transmission selection can coexist with priorities using 802.1Qav queuing for time-sensitive streams.

Data Center Bridging eXchange (DCBX) Protocol

Data Center Bridging Exchange (DCBX) Protocol allows auto-exchange of Ethernet parameters and discovery functions between switches and endpoints. DCBX includes data center Ethernet peer discovery, mismatched configuration detection, and data center Ethernet link configuration of peers.

Quantized Congestion Notification (QCN) Protocol

QCN is a IEEE 802.1Qau mechanism that manages network congestion. When a queue reaches a configured threshold, QCN throttles traffic at the source of the congestion by transmitting messages that propagate back to the source and temporarily stop the source from transmitting. When the queue crosses the threshold that indicates the congestion has dissipated, QCN sends a message to allow the source to resume transmitting frames. QCN includes the following commands:

Software Overview

Figure 1-1 below provides an overview of DCB feature that is a part of the NSM Layer 2 module and its relationship with the other modules in ZebOS-XP. DCB uses most of the NSM services and communicates to hardware through HAL/HSL layers.

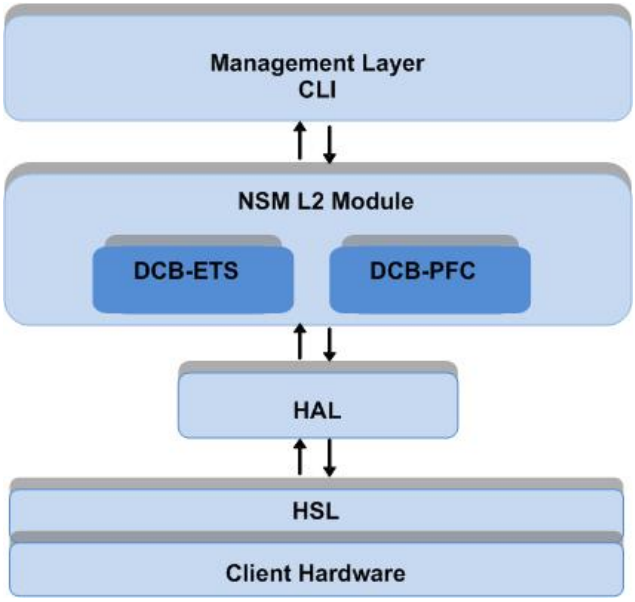

Figure 1-1: DCB-ETS System Architecture

Management Layer (CLI)

The Command Line Interfacer (CLI) is a text-based facility. You can use many of the commands in scripts to automate configuration tasks. Each CLI command is usually associated with a specific function or a common function performing a specific task. The IMI (Integrated Management Interface) Shell gives users and administrators the ability to issue commands to several daemons from a single TELNET session.

NSM Layer 2 Module

Both the DCB-ETS and DCB-PFC feature are a part of NSM layer 2 module to provide an association with rest of the modules within ZebOS-XP.

HAL

A hardware abstraction layer (HAL) is an abstraction layer implemented in the software between the physical hardware of a computer and the software that runs on that computer.

HSL

The ZebOS-XP Hardware Services Layer (HSL) is an array of software modules; each designed and integrated with an industry-leading merchant silicon solution and operating system. HSL provides a comprehensive forwarding plane implementation supporting Layer 2, Layer 3 (both IPv4 and IPv6), multicast and MPLS/Traffic Engineering.

Software Design

This document includes modules to configure PFC, ETS, and Application Priority per end station or bridge port.

PFC Configuration Manager

This PFC Configuration Manager module implements the configuration commands and messaging to HAL module with the DCB-PFC related parameters. It supports the CLI commands and corresponding API to provide the following functionality:

- Enable or disable the PFC at switch and interface level (for DCBX domain).
- Set or unset the PFC mode.
- Set the PFC cap.
- Set or unset “PFCLinkDelayAllowance” (that is, the allowance made for round-trip propagation delay of the link in bits).
- Display PFC requests sent and indications received.

ETS Configuration Manager

The ETS configuration manager implements the configuration commands and messaging to the HAL module with the related DCB-ETS parameters. It supports the CLI commands and corresponding API to provide the following functionality:

- Enable or disable the ETS features at the switch or interface level.
- Add or remove the priorities in a Traffic Class Group (TCG).
- Assign bandwidth percentages to a TCG.
- Remove a TCG configuration from an interface.
- Display information on the Traffic Class Groups.

Application-priority Configuration Manager

The Application-priority configuration manager module implements the configuration commands and messaging to the HAL module to provide the application-priority related parameters. It supports the CLI commands and corresponding API to provide the following functionality:

- Set or unset the application priority
- Display application priority configuration

CHAPTER 2 Data Structures

This chapter lists all of the data structures used with the ZebOS-XP modules.

Common Data Structures

See the *Common Data Structures Developer Guide* for a description of these data structures used by multiple ZebOS-XP modules:

- `cli`
- `interface`
- `lib_globals`
- `nsm_bridge`
- `nsm_bridge_master`

lldp_master

This data structure contains the configuration parameters and variables used with the LLDP master. It is defined in the `onmd/lldp/lldpd.h` file.

Member	Description
*lldp_rcv_thread	Receive and process LLDP frames
*lldp_if_list	List of LLDP interfaces
if_cnt	Count: Number of LLDP interfaces enabled.
syscap	Integer signifying system capabilities. • If least significant bit is set: IPV4_ROUTING • If next to LSB is set: IPV6_ROUTING • If third bit from LSB is set: L2_SWICHING
sys_cap_enabled	Signifies system capabilities enabled in the system
lldp_stats_rem_drops	Number of LLDP stats table drops
lldp_stats_rem_inserts	Number of LLDP table inserts
lldp_stats_rem_deletes	Number of lldp table deletes
sys_name[LLDP_NAME_MAX_SIZE	System name
sys_descr[LLDP_DESCR_MAX_SIZE + 1]	System description

Member	Description
lldp_stats_rem_last_change_time	Time when LLDP table was modified
lldp_dest_addr[ETHER_ADDR_LEN	Address of LLDP destination
pal_in4_addr mgmt_addr	Management address
conf_flag	Flag to signify whether LLDP management address is configured or not
bool_t fast_init	This is used to indicate fast-init is either enabled or disabled.

Definition

```
struct lldp_master
{
  struct onmd_master *onm;

  pal_sock_handle_t sockfd;
  struct thread *lldp_rcv_thread;

  struct list *lldp_if_list;

  u_int16_t  if_cnt;
  u_int16_t  syscap;
  u_int16_t  sys_cap_enabled;
  u_int32_t  lldp_stats_rem_drops;
  u_int32_t  lldp_stats_rem_inserts;
  u_int32_t  lldp_stats_rem_deletes;
  u_char sys_name[LLDP_NAME_MAX_SIZE + 1];
  u_char sys_descr[LLDP_DESCR_MAX_SIZE + 1];
  pal_time_t lldp_stats_rem_last_change_time;
  u_char     lldp_dest_addr[ETHER_ADDR_LEN];
  struct pal_in4_addr mgmt_addr;

#define LLDP_CONF_MGMT_IP_ADDRESS    (1 << 0)
  u_int8_t conf_flag;

#ifdef HAVE_DCB

  /*This is to indicate fast-init enable/disable*/
  bool_t fast_init;
#endif /* HAVE_DCB */
};
```

nsm_dcb_if

This data structure contains the configuration information for the DCB interface. It is defined in the `nsm/L2/nsm_dcb.h` file.

Member	Description
nsm_dcb_bridge *dcbg	Pointer to the DCB bridge.
interface *ifp;	Interface pointer.
nsm_qcn_cnpv_data	This structure contains details about CNPV.
nsm_qcn_cnpv_data[NSM_NUM_CNPV	Array of CNPV data structure.
is_cp	Flag indicating if CP is enabled.
pfc_config *pfc_config_admin;	Admin parameters.
pfc_config *pfc_config_oper	Operational parameters.
nsm_dcb_mode pfc_mode	
ets_config *ets_config_admin	Admin parameters.
ets_config *ets_config_oper	Operational parameters.
ets_rec *ets_rec_param	Recommended parameters.
nsm_dcb_mode ets_mode	Enum that contains ETS mode on/auto
app_config *app_config_admin	Admin parameters.
app_config *app_config_oper	Operational parameters.
nsm_dcb_mode app_mode	This indicates the DCB flags on interface.
pfc_requests_sent	This is to maintain number of PFC requests sent.
pfc_indications_rcvd	This is to maintain number of PFC indications received.
avl_tree *dcb_appl_pri	This will be list of all application priorities set on the interface.

Definition

```
/* Pointer to dcb bridge */
  struct nsm_dcb_bridge *dcbg;

  /* Interface pointer. */
  struct interface *ifp;

  struct nsm_qcn_cnpv_data nsm_qcn_cnpv_data[NSM_NUM_CNPV];
  bool_t is_cp;

  /* Priority Flow Control Configuration for the interface */
  struct pfc_config *pfc_config_admin; /* Admin params */
  struct pfc_config *pfc_config_oper; /* Operational Params */
  enum nsm_dcb_mode pfc_mode;

  /* ETS configuration for the interface */
```

```
  struct ets_config *ets_config_admin; /* Admin Params*/
  struct ets_config *ets_config_oper;  /* Operational Params */
  struct ets_rec *ets_rec_param;       /* Recommendational Params */
  enum nsm_dcb_mode ets_mode;


  /* Application Priority configuration on the interface */
  struct app_config *app_config_admin; /* Admin Params */
  struct app_config *app_config_oper; /* Operational Params */
  enum nsm_dcb_mode app_mode;
  /* This indicates the DCB flags on interface */
  u_int8_t dcb_if_flags;
#define NSM_DCB_IF_ENABLE (1 << 0)
#define NSM_DCB_IF_ETS_ENABLE (1 << 1)
#define NSM_DCB_IF_PFC_ENABLE (1 << 2)

  /* This is to maintain number of PFC requests sent*/
   u_int32_t pfc_requests_sent;
 /* This is to maintain number of PFC indications received*/
   u_int32_t pfc_indications_rcvd;
  /* This will be list of all application priorities set on the interface */
  struct avl_tree *dcb_appl_pri;
};
```

nsm_hal_qcn_data

This data structure is defined in the `nsm/L2/nsm_dcb.h` file.

Member	Description
master_enable	Flag indicating whether QCN is enable or disable.
cnm_transmit_priority	Contains the CNM transmit priority.
discarded_frames	Total discarded frames across all CP.
err_port_cnt	Number of ports present in err_port_list.
*err_port_list[10]	A list of Ports whose alternate priority values specify a CNPV.

Definition

```
struct nsm_hal_qcn_data
{
  bool master_enable;
  int cnm_transmit_priority;
  int discarded_frames;
  int err_port_cnt;
  char *err_port_list[10];
```

```
};
```

nsm_hal_cp_if_data

This data structure is defined in the nsm/L2/nsm_dcb.h file.

Member	Description
interface *ifp	This is main Interface pointer.
cp_count	Number of CP configured
nsm_hal_cp_data cp_data[NSM_NUM_CNPV]	List containing data related to all CPs

Definition

```
struct nsm_hal_cp_if_data
{
  struct interface *ifp;
  int cp_count;
  struct nsm_hal_cp_data cp_data[NSM_NUM_CNPV];
};
```

nsm_hal_cp_data

This data structure is defined in the nsm/L2/nsm_dcb.h file.

Member	Description
ifname	Interface name
cp_mac_addr [ETHER_ADDR_LEN]	MAC address belonging to the system transmitting the CNM PDU
cp_id	A number uniquely identifying a CP
qsp	Set-point for the CP's queue.
qlen	Current number of octets in the CP’s queue
qlenold	Previous value of QLen
qoffset	Required to calculate the quantized feedback field
qdelta	Required to calculate the quantized feedback field
fb	Quantized feedback field
enqued	Number of octets remaining to be enqueued by CP before a CNM PDU is generated

Member	Description
samplebase	Minimum number of octets to enqueue in the CP's queue between CNM PDU transmissions
transmitted_frames	Number of data frames enqueued for transmission on this CP's output queue
minhdroctet	Minimum number of octets that CP returns in the CNM
transmitted_cnms	Number of CNMs transmitted by CP
discarded_frames	Number of frames discarded by this CP because of full queue

Definition

```
struct nsm_hal_cp_data
{
  char *ifname;
  u_char cp_mac_addr [ETHER_ADDR_LEN];
  u_int32_t cp_id;
  u_int32_t qsp;
  u_int32_t qlen;
  u_int32_t qlenold;
  float weight;
  s_int32_t qoffset;
  s_int32_t qdelta;
  s_int32_t fb;
  s_int32_t enqued;
  u_int32_t samplebase;
  u_int32_t transmitted_frames;
  u_int32_t minhdroctet;
  u_int32_t transmitted_cnms;
  u_int32_t discarded_frames;
};
```

CHAPTER 3 DCB-ETS Functions

This chapter contains the APIs for DCB-ETS (Enhanced Transmission Selection). It includes the following APIs:

Function	Description
hal_dcb_init	Sends a HAL_MSG_DCB_INIT message to HSL to initialize DCB in the hardware.
hal_dcb_deinit	Sends a HAL_MSG_DCB_DEINIT message to HSL to deinitialize DCB in the hardware.
hal_dcb_bridge_enable	Sends a HAL_MSG_DCB_ENABLE message to HSL to enable DCB in the hardware.
hal_dcb_bridge_disable	Sends a HAL_MSG_DCB_DISABLE message to HSL to disable DCB in the hardware.
hal_dcb_ets_bridge_enable	Sends a HAL_MSG_DCB_ETS_ENABLE message to HSL to enable ETS in the hardware.
hal_dcb_ets_bridge_disable	Sends a HAL_MSG_DCB_ETS_DISABLE message to HSL to enable ETS in the hardware
hal_dcb_interface_enable	Sends a HAL_MSG_DCB_IF_ENABLE message to HSL to enable DCB in the hardware on the interface.
hal_dcb_interface_disable	Sends a HAL_MSG_DCB_IF_DISABLE message to HSL to disable DCB in the hardware on the interface.
hal_dcb_ets_interface_enable	Sends a HAL_MSG_DCB_ETS_IF_ENABLE message to HSL to enable the ETS feature of DCB in the hardware on the interface
hal_dcb_ets_interface_disable	Sends a HAL_MSG_DCB_ETS_IF_DISABLE message to HSL to disable the ETS feature of DCB in the hardware on the interface.
hal_dcb_select_ets_mode	Sends a HAL_MSG_DCB_ETS_SELECT_ETS_MODE message to HSL.
hal_dcb_ets_set_application_priority	Sends a HAL_MSG_DCB_ETS_TCG_APP_PRIO_SET message to HSL.
hal_dcb_ets_unset_application_priority	Sends a HAL_MSG_DCB_ETS_TCG_APP_PRIO_UNSET message to HSL.
hal_dcb_ets_assign_bw_to_tcgs	Sends a HAL_MSG_DCB_ETS_TCG_BW_SET message to HSL.
hal_dcb_ets_add_pri_to_tcg	Sends a HAL_MSG_DCB_ETS_ADD_PRI_TO_TCG message to HSL.
hal_dcb_ets_remove_pri_from_tcg	Removes a HAL_MSG_DCB_ETS_ADD_PRI_TO_TCG message from HSL.
hal_dcb_global_disable	Sends a HAL_MSG_DCB_DISABLE message to HSL to disable DCB at the switch level in the hardware.
lldp_dcbx_set_capability	Enables DCBX on an interface.
lldp_set_fast_init	Enables fast initialization on LLDP.

Function	Description
nsm_dcb_add_pri_to_tcg	Adds priorities to a traffic class group.
nsm_dcb_remove_pri_from_tcg	Removes priorities from a traffic class group.
nsm_dcb_max_tcg	Sets the maximum number of traffic class groups that can be configured on an interface.
nsm_dcb_assign_bw_percentage_to_tcg	Assigns the bandwidth to the traffic class groups.
nsm_dcb_delete_tcgs	Deletes the TCG and related configuration from hardware.
nsm_dcb_app_bridge_enable	Enables application priority for a switch.
nsm_dcb_app_bridge_disable	Disables application priority from a switch.
nsm_dcb_app_enable_interface	Enables application priority for an interface.
nsm_dcb_app_disable_interface	Disables application priority for an interface.
nsm_dcb_app_set_advertise_interface	Enables advertising application priority for an interface.
nsm_dcb_application_priority_set	Sets the application priority based on the interface for the given protocol ID.
nsm_dcb_application_priority_unset	Unsets the application priority based on the interface from a given protocol ID.
nsm_dcb_bridge_disable	Disables DCB from a switch.
nsm_dcb_bridge_enable	Enables DCB on a switch.
nsm_dcb_init	Initializes the `nsm_dcb_global` structure.
nsm_dcb_deinit	De-initializes the `nsm_dcb_global` structure.
nsm_dcb_enable_interface	Enables the DCB features on an interface.
nsm_dcb_disable_interface	Disables the DCB features from an interface.
nsm_dcb_ets_bridge_enable	Enables ETS on a switch (bridge).
nsm_dcb_ets_bridge_disable	Disables ETS on a switch (bridge).
nsm_dcb_ets_set_willing_interface	Enables the “willing” setting for ETS on an interface.
nsm_dcb_ets_set_advertise_interface	Enables the “advertise” setting for ETS on an interface.
nsm_dcb_show_appl_priority_table	Displays the application priority table for a given interface.
nsm_dcb_show_tcg_by_bridge	Displays all traffic class groups for a given bridge.
nsm_dcb_show_tcg_by_intf	Displays all traffic class groups for a given interface.
nsm_dcb_show_app_by_bridge	Displays application priority information configured on a given bridge.

hal_dcb_init

This function sends a `HAL_MSG_DCB_INIT` message to HSL to initialize DCB in the hardware. This function performs the basic initialization that is required by the hardware for the DCB feature.

Syntax

```
s_int32_t hal_dcb_init (char *bridge_name)
```

Input Parameters

`bridge_name` Bridge name

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_deinit

This function sends a `HAL_MSG_DCB_DEINIT` message to HSL to deinitialize DCB in the hardware.

Syntax

```
s_int32_t hal_dcb_deinit (char *bridge_name)
```

Input Parameters

`bridge_name` Bridge name

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_bridge_enable

This function sends a `HAL_MSG_DCB_ENABLE` message to HSL to enable DCB in the hardware.

Syntax

```
s_int32_t
hal_dcb_bridge_enable (char *bridge_name)
```

Input Parameters

`bridge_name` Bridge ID

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_bridge_disable

This function sends a `HAL_MSG_DCB_DISABLE` message to HSL to disable DCB in the hardware.

Syntax

```
s_int32_t
hal_dcb_bridge_disable (char *bridge_name)
```

Input Parameters

`bridge_name` Bridge ID

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_ets_bridge_enable

This function sends a `HAL_MSG_DCB_ETS_ENABLE` message to HSL to enable ETS in the hardware.

Syntax

```
s_int32_t
hal_dcb_ets_bridge_enable (char *bridge_name)
```

Input Parameters

`bridge_name` Bridge ID

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_ets_bridge_disable

This function sends a `HAL_MSG_DCB_ETS_DISABLE` message to HSL to enable ETS in the hardware.

Syntax

```
s_int32_t
hal_dcb_ets_bridge_disable (char *bridge_name)
```

Input Parameters

`bridge_name` Bridge ID

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_interface_enable

This function sends a `HAL_MSG_DCB_IF_ENABLE` message to HSL to enable DCB in the hardware on the interface. The message passes the interface index and default configuration parameters to HSL.

Syntax

```
s_int32_t hal_dcb_interface_enable (char *bridge_name, s_int32_t ifindex)
```

Input Parameters

`bridge_name`	Bridge name
`ifindex`	Interface index

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_interface_disable

This function sends a `HAL_MSG_DCB_IF_DISABLE` message to HSL to disable DCB in the hardware on the interface. The message passes the interface index.

Syntax

```
s_int32_t hal_dcb_interface_disable (char *bridge_name, s_int32_t ifindex)
```

Input Parameters

`bridge_name`	Bridge name
`ifindex`	Interface index

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_ets_interface_enable

This function sends a `HAL_MSG_DCB_ETS_IF_ENABLE` message to HSL to enable the ETS feature of DCB in the hardware on the interface. The message passes the interface index and default configuration parameters to HSL.

Syntax

```
s_int32_t hal_dcb_ets_interface_enable (char *bridge_name, s_int32_t ifindex)
```

Input Parameters

`bridge_name`	Bridge name
`ifindex`	Interface index

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_ets_interface_disable

This function sends a `HAL_MSG_DCB_ETS_IF_DISABLE` message to HSL to disable the ETS feature of DCB in the hardware on the interface. The message passes the interface index to HSL.

Syntax

```
s_int32_t
hal_dcb_ets_interface_disable (char *bridge_name, s_int32_t ifindex)
```

Input Parameters

`bridge_name`	Bridge name
`ifindex`	Interface index

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_select_ets_mode

This function sends a `HAL_MSG_DCB_ETS_SELECT_ETS_MODE` message to HSL, so that HSL can set or unset the willing bit in the ETS configuration table based on the mode. The message passes the interface index and mode value.

Syntax

```
s_int32_t
hal_dcb_select_ets_mode (char *bridge_name, s_int32_t ifindex,
                         enum hal_dcb_mode mode)
```

Input Parameters

`bridge_name`	Bridge name
`ifindex`	Interface index

mode	Whether willing is enabled or disabled

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_ets_set_application_priority

This function sends a HAL_MSG_DCB_ETS_TCG_APP_PRIO_SET message to HSL, so that HSL updates the application priority table in the hardware. The message passes the interface index, protocol selector, protocol ID, and priority.

Syntax

```
s_int32_t
hal_dcb_ets_set_application_priority (char *bridge_name, s_int32_t ifindex,
                                      u_int8_t sel, u_int16_t proto_id,
                                      u_int8_t pri)
```

Input Parameters

bridge_name	Bridge name
ifindex	Interface index
sel	Protocol selector
proto_id	Protocol ID.
pri	Priority

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_ets_unset_application_priority

This function sends a HAL_MSG_DCB_ETS_TCG_APP_PRIO_UNSET message to HSL, so that HSL updates the application priority table in the hardware. The message passes the interface index, protocol selector, protocol ID, and priority.

Syntax

```
s_int32_t
hal_dcb_ets_unset_application_priority (char *bridge_name, s_int32_t ifindex,
                                        u_int8_t sel, u_int16_t proto_id,
                                        u_int8_t pri)
```

Input Parameters

bridge_name	Bridge name

`ifindex`	Interface index
`sel`	Protocol selector
`proto_id`	Protocol ID.
`pri`	Priority

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_ets_assign_bw_to_tcgs

This function sends a `HAL_MSG_DCB_ETS_TCG_BW_SET` message to HSL, so that HSL updates the traffic class group and bandwidth assignment table in the hardware. The message passes the interface index and bandwidth assignment array of all traffic class groups.

Syntax

```
s_int32_t
hal_dcb_ets_assign_bw_to_tcgs (char *bridge_name, s_int32_t ifindex,
                               u_int8_t *bw)
```

Input Parameters

`bridge_name`	Bridge name
`ifindex`	Interface index
`bw`	Bandwidth value

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_ets_add_pri_to_tcg

This function sends a `HAL_MSG_DCB_ETS_ADD_PRI_TO_TCG` message to HSL, so that HSL updates the traffic class group and priority assignment table in the hardware. The message passes the interface index, TCG ID, and priority.

Syntax

```
s_int32_t
hal_dcb_ets_add_pri_to_tcg (char *bridge_name, s_int32_t ifindex,
                            u_int8_t tcgid, u_int8_t pri)
```

Input Parameters

`bridge_name`	Bridge name
`ifindex`	Interface index

`tcgid`	Traffic Class Group Identifier
`pri`	Priority

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_ets_remove_pri_from_tcg

This function removes a `HAL_MSG_DCB_ETS_ADD_PRI_TO_TCG` message from HSL.

Syntax

```
s_int32_t
hal_dcb_ets_remove_pri_from_tcg (char *bridge_name, s_int32_t ifindex,
                                 u_int8_t tcgid, u_int8_t pri)
```

Input Parameters

`bridge_name`	Bridge name
`ifindex`	Interface index
`tcgid`	Traffic Class Group Identifier
`pri`	Priority

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_global_disable

This function sends a `HAL_MSG_DCB_DISABLE` message to HSL to disable DCB at the switch level in the hardware.

Syntax

```
s_int32_t
hal_dcb_global_disable (char *bridge_name)
```

Input Parameters

`bridge_name`	Bridge name

Output Parameters

None

Return

-1

0

lldp_dcbx_set_capability

This function enables DCBX on an interface.

Syntax

```
s_int32_t
lldp_dcbx_set_capability (struct interface *ifp, bool_t state)
```

Input Parameters

`ifp`	Interface pointer
`state`	Whether DCBX is enabled or disabled:
`PAL_TRUE`	Enabled
`PAL_FALSE`	Disabled

Output Parameters

None

Return Values

LLDP_API_SUCCESS when the function succeeds

LLDP_API_ERR_ONM_IF_NOT_EXIST when the `onmd` interface does not exist

LLDP_API_ERR_LLDP_IF_NOT_EXIST when the LLDP interface does not exist

LLDP_API_ERR_AGG_IF when the interface is aggregated

LLDP_API_ERR_DCBX_ENABLE_NO_RXTX when receive transmission is not enabled

LLDP_API_ERR_DCBX_ENABLE_EXIST when DCBX is already enabled

LLDP_API_ERR_DCBX_ENABLE_NOT_EXIST when DCBX is already disabled

lldp_set_fast_init

This function enables fast initialization on LLDP.

Syntax

```
s_int32_t
lldp_set_fast_init (struct lldp_master *lldpm, bool_t state)
```

Input Parameters

`lldpm`	LLDP master
`state`	Whether LLDP fast initialization is enabled or disabled:
`PAL_TRUE`	Enabled
`PAL_FALSE`	Disabled

Output Parameters

None

Return Values

LLDP_API_SUCCESS when the function succeeds

LLDP_API_ERR_LLDP_MASTER_NOT_EXIST when `lldpm` is NULL

nsm_dcb_add_pri_to_tcg

This function adds priorities to a traffic class group.

Syntax

```
s_int32_t
nsm_dcb_add_pri_to_tcg (u_int32_t vr_id, char *ifname, u_int8_t tcgid,
                        u_int8_t pri)
```

Input Parameters

`vr_id`	Virtual Router ID
`ifname`	Interface name
`tcgid`	Traffic Class Group Identifier
`pri`	Priority

Output Parameters

None

Return Values

NSM_DCB_API_SET_ERR_WRONG_TCG

NSM_DCB_API_SET_ERR_TCG_PRI_EXISTS

NSM_DCB_API_SET_SUCCESS

nsm_dcb_remove_pri_from_tcg

This function removes priorities from a traffic class group.

Syntax

```
s_int32_t
nsm_dcb_remove_pri_from_tcg (u_int32_t vr_id, char *ifname, u_int8_t tcgid,
                             u_int8_t pri)
```

Input Parameters

`vr_id`	Virtual Router ID
`ifname`	Interface name
`tcgid`	Traffic Class Group Identifier
`pri`	Priority

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS

nsm_dcb_max_tcg

This function sets the maximum number of traffic class groups that can be configured on an interface.

Syntax

```
s_int32_t
nsm_dcb_max_tcg (u_int32_t vr_id, char *ifname, u_int8_t maxtcg)
```

Input Parameters

`vr_id`	Virtual Router ID.
`ifname`	Interface name.
`maxtcg`	The maximum number of traffic class groups.

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found.

NSM_DCB_API_SET_ERR_INTERFACE if the interface is NULL.

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found.

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found.

NSM_DCB_API_SET_ERR_INTERFACE_NO_ETS when ETS is not enabled on the interface.

nsm_dcb_assign_bw_percentage_to_tcg

This function assigns the bandwidth to the traffic class groups. The total of bandwidth should be 100.

Syntax

```
s_int32_t nsm_dcb_assign_bw_percentage_to_tcg (u_int32_t vr_id, char *ifname, u_int16_t
*bw)
```

Input Parameters

`vr_id`	Virtual Router ID
`ifname`	Interface name
`bw`	Bandwidth value

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

nsm_dcb_delete_tcgs

This function deletes the TCG and related configuration from hardware.

Syntax

```
s_int32_t nsm_dcb_delete_tcgs (u_int32_t vr_id, char *ifname)
```

Input Parameters

`vr_id`	Virtual Router ID
`ifname`	Interface name

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

nsm_dcb_app_bridge_enable

This function enables application priority for a switch (bridge).

Syntax

```
s_int32_t
nsm_dcb_app_bridge_enable (u_int32_t vr_id, char *bridge_name)
```

Input Parameters

`vr_id`	Virtual Router ID
`bridge_name`	Bridge ID

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found.

NSM_DCB_API_SET_ERR_INTERFACE if the interface is NULL.

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found.

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found.

NSM_DCB_API_SET_ERR_NO_VLAN when the bridge is not VLAN-aware

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found or DCB is not enabled.

NSM_DCB_API_SET_ERR_APP_EXISTS when application priority is already enabled for the bridge.

nsm_dcb_app_bridge_disable

This function disables application priority for a switch (bridge).

Syntax

```
s_int32_t
nsm_dcb_app_bridge_disable (u_int32_t vr_id, char *bridge_name)
```

Input Parameters

`vr_id`	Virtual Router ID
`bridge_name`	Bridge ID.

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found

NSM_DCB_API_SET_ERR_INTERFACE if the interface is NULL

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found

NSM_DCB_API_SET_ERR_NO_APP when application priority is not enabled.

nsm_dcb_app_enable_interface

This function enables application priority for an interface.

Syntax

```
s_int32_t
nsm_dcb_app_enable_interface (u_int32_t vr_id, char *ifname, bool_t mode)
```

Input Parameters

`vr_id`	Virtual Router ID
`ifname`	Interface name
`mode`	Reserved for future use

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found.

NSM_DCB_API_SET_ERR_INTERFACE if the interface is NULL.

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found.

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found.

NSM_DCB_API_SET_ERR_NO_APP when application priority is not enabled for the switch.

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found.

NSM_DCB_API_SET_ERR_INTERFACE_NO_ETS when ETS is not enabled on the interface.

NSM_DCB_API_SET_ERR_APP_EXISTS when application priority is already enabled for the interface.

NSM_DCB_API_SET_ERR_NO_MEM when there is an error allocating memory

nsm_dcb_app_disable_interface

This function disables application priority for an interface.

Syntax

```
s_int32_t
nsm_dcb_app_disable_interface (u_int32_t vr_id, char *ifname)
```

Input Parameters

`vr_id`	Virtual Router ID
`ifname`	Interface name

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found.

NSM_DCB_API_SET_ERR_INTERFACE if the interface is NULL.

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found.

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found.

NSM_DCB_API_SET_ERR_NO_APP when application priority is not enabled.

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found.

NSM_DCB_API_SET_ERR_INTERFACE_NO_APP when application priority is already disabled for the interface.

nsm_dcb_app_set_advertise_interface

This function enables advertising application priority for an interface.

Syntax

```
s_int32_t
nsm_dcb_app_set_advertise_interface (u_int32_t vr_id, char *ifname,
                                     bool_t advertise)
```

Input Parameters

`vr_id`	Virtual Router ID
`ifname`	Interface name

`advertise`	Whether to enable or disable advertising

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found.

NSM_DCB_API_SET_ERR_INTERFACE if the interface is NULL.

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found.

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found.

NSM_DCB_API_SET_ERR_NO_APP when application priority is not enabled for the bridge.

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found.

NSM_DCB_API_SET_ERR_INTERFACE_NO_APP when application priority is not enabled for the interface.

nsm_dcb_application_priority_set

This function sets the application priority based on the interface for the given protocol ID.

Syntax

```
s_int32_t
nsm_dcb_application_priority_set (u_int32_t vr_id, char *ifname, u_int8_t sel,
                                  u_int16_t proto_id, u_int8_t prio_map,
                                  enum dcb_appl_pri_mapping_type type)
```

Input Parameters

`vr_id`	Virtual Router ID
`ifname`	Interface name.
`sel`	Protocol selector
`proto_id`	Protocol ID.
`prio_map`	Priority map
`type`	Protocol type.

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

nsm_dcb_application_priority_unset

This function unsets the application priority based on the interface for the given protocol ID.

Syntax

```
s_int32_t
nsm_dcb_application_priority_unset (u_int32_t vr_id, char *ifname, u_int8_t sel,
                                    u_int16_t proto_id, u_int8_t prio_map,
                                    enum dcb_appl_pri_mapping_type type)
```

Input Parameters

`vr_id`	Virtual Router ID
`ifname`	Interface name.
`sel`	Protocol selector
`proto_id`	Protocol ID.
`prio_map`	Priority map
`type`	Protocol type.

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

NSM_DCB_API_SET_ERR_NO_APP_PRIO

nsm_dcb_bridge_disable

This function disables DCB on a switch (bridge).

Syntax

```
s_int32_t
nsm_dcb_bridge_disable (u_int32_t vr_id, char *bridge_name)
```

Input Parameters

`vr_id`	Virtual Router ID
`bridge_name`	Bridge ID. If NULL, this function disables DCB on the default bridge.

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found.

NSM_BRIDGE_ERR_NOTFOUND when the given bridge or the default bridge is not found.

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found.

NSM_DCB_API_SET_ERR_DCB_EXISTS when DCB is already disabled.

NSM_DCB_API_SET_ERR_HW_NO_SUPPORT when DCB cannot be disabled in the hardware.

nsm_dcb_bridge_enable

This function enables DCB on a switch (bridge).

Syntax

```
s_int32_t
nsm_dcb_bridge_enable (u_int32_t vr_id, char *bridge_name)
```

Input Parameters

`vr_id`	Virtual Router ID
`bridge_name`	Bridge ID. If NULL, this function enables DCB on the default bridge.

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found.

NSM_BRIDGE_ERR_NOTFOUND when the given bridge or the default bridge is not found.

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found.

NSM_DCB_API_SET_ERR_DCB_EXISTS when DCB is already enabled.

NSM_DCB_API_SET_ERR_HW_NO_SUPPORT when DCB cannot be enabled in the hardware.

nsm_dcb_init

This function initializes the `nsm_dcb_global` structure. It allocates memory for the DCB global structure and creates the AVL tree for DCB interfaces.

Syntax

```
s_int32_t
nsm_dcb_init (struct nsm_bridge *bridge)
```

Input Parameters

`bridge`	Bridge on which DCB should be initiated.

Output Parameters

None

Return Values

NULL

nsm_dcb_deinit

This function deinitializes the `nsm_dcb_global` structure. It frees in the AVL tree for interfaces and frees the memory assigned to global DCB structure.

Syntax

```
s_int32_t
nsm_dcb_deinit (struct nsm_bridge_master *master)
```

Input Parameters

`master`	Pointer to the master structure

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

nsm_dcb_enable_interface

This function enables the DCB features on the interface.

Syntax

```
s_int32_t
nsm_dcb_enable_interface (u_int32_t vr_id, char *ifname)
```

Input Parameters

`vr_id`	Virtual Router ID
`ifname`	Interface name

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

NSM_DCB_API_SET_ERR_HW_NO_SUPPORT

NSM_DCB_API_SET_ERR_DCB_INTERFACE

NSM_BRIDGE_ERR_NOTFOUND

NSM_DCB_API_SET_ERR_NO_DCBG

NSM_DCB_API_SET_ERR_NO_DCB

nsm_dcb_disable_interface

This function disables the DCB features on the interface.

Syntax

```
s_int32_t
nsm_dcb_disable_interface (u_int32_t vr_id, char *ifname)
```

Input Parameters

`vr_id`	Virtual Router ID

`ifname`	Interface name

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

nsm_dcb_ets_bridge_enable

This function enables ETS on a switch (bridge).

Syntax

```
s_int32_t
nsm_dcb_ets_bridge_enable (u_int32_t vr_id, char *bridge_name)
```

Input Parameters

`vr_id`	Virtual Router ID
`bridge_name`	Bridge ID. If NULL, this function enables ETS on the default bridge.

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found.

NSM_BRIDGE_ERR_NOTFOUND when the given bridge or the default bridge is not found.

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled.

NSM_DCB_API_SET_ERR_NO_VLAN when the bridge is not VLAN-aware.

NSM_DCB_API_SET_ERR_ETS_EXISTS when ETS is already enabled.

nsm_dcb_ets_bridge_disable

This function disables ETS on a switch (bridge).

Syntax

```
s_int32_t
nsm_dcb_ets_bridge_disable (u_int32_t vr_id, char *bridge_name)
```

Input Parameters

`vr_id`	Virtual Router ID
`bridge_name`	Bridge ID. If NULL, this function disables ETS on the default bridge.

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found.

NSM_BRIDGE_ERR_NOTFOUND when the given bridge or the default bridge is not found.

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled.

NSM_DCB_API_SET_ERR_NO_VLAN when the bridge is not VLAN-aware.

NSM_DCB_API_SET_ERR_INTERFACE_NO_ETS when ETS is not enabled.

NSM_DCB_API_SET_ERR_HW_NO_SUPPORT when ETS cannot be disabled in the hardware.

NSM_DCB_API_SET_ERR_ETS_INTERFACE when there is an internal error.

nsm_dcb_ets_enable_interface

This function enable ETS feature of DCB on the interface.

Syntax

```
s_int32_t
nsm_dcb_ets_enable_interface (u_int32_t vr_id, char *ifname, bool_t mode)
```

Input Parameters

`vr_id`	Virtual Router ID
`ifname`	Interface name
`mode`	One of these constants from the `nsm_dcb_mode` enum in `nsm/L2/nsm_dcb.h`:

`NSM_DCB_MODE_ON`

`NSM_DCB_MODE_AUTO`

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

nsm_dcb_ets_disable_interface

This function disables the ETS on an interface.

Syntax

```
s_int32_t
nsm_dcb_ets_disable_interface (u_int32_t vr_id, char *ifname)
```

Input Parameters

`ifname`	Interface name
`vr_id`	Virtual Router ID

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

nsm_dcb_ets_set_willing_interface

This function enables the “willing” setting for ETS on an interface.

Syntax

```
s_int32_t
nsm_dcb_ets_set_willing_interface (u_int32_t vr_id, char *ifname,
                                   bool_t willing)
```

Input Parameters

`vr_id`	Virtual Router ID
`ifname`	Interface name.
`willing`	Whether willing is enabled is disabled.

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found.

NSM_DCB_API_SET_ERR_INTERFACE if the interface is NULL.

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found.

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled.

NSM_DCB_API_SET_ERR_NO_ETS when ETS is not enabled.

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found.

NSM_DCB_API_SET_ERR_INTERFACE_NO_ETS when ETS is not enabled on the interface.

nsm_dcb_ets_set_advertise_interface

This function enables the “advertise” setting for ETS on an interface.

Syntax

```
s_int32_t
nsm_dcb_ets_set_advertise_interface (u_int32_t vr_id, char *ifname,
                                     bool_t advertise)
```

Input Parameters

`vr_id`	Virtual Router ID

`ifname`	Interface name.
`advertise`	Whether advertise is enabled is disabled.

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found.

NSM_DCB_API_SET_ERR_INTERFACE if the interface is NULL.

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found.

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled.

NSM_DCB_API_SET_ERR_NO_ETS when ETS is not enabled.

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found.

NSM_DCB_API_SET_ERR_INTERFACE_NO_ETS when ETS is not enabled on the interface.

nsm_dcb_show_appl_priority_table

This function displays the application priority table for a given interface.

Syntax

```
s_int32_t
nsm_dcb_show_appl_priority_table (struct cli *cli, char *ifname)
```

Input Parameters

`cli`	CLI structure
`ifname`	Interface name

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found.

NSM_DCB_API_SET_ERR_INTERFACE if the interface is NULL.

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found.

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled.

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found.

NSM_DCB_API_SET_ERR_INTERFACE_NO_DCB when DCB is not enabled on the interface.

nsm_dcb_show_tcg_by_bridge

This function displays all traffic class groups for a given bridge.

Syntax

```
s_int32_t
nsm_dcb_show_tcg_by_bridge (struct cli *cli, char *bridge_name)
```

Input Parameters

`cli`	CLI structure
`bridge_name`	Bridge ID

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found.

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found.

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled.

NSM_DCB_API_SET_ERR_NO_ETS when ETS is not enabled.

nsm_dcb_show_tcg_by_intf

This function displays all traffic class groups for a given interface.

Syntax

```
s_int32_t
nsm_dcb_show_tcg_by_intf (struct cli *cli, char *ifname)
```

Input Parameters

`cli`	CLI structure
`ifname`	Interface name

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found.

NSM_DCB_API_SET_ERR_INTERFACE if the interface is NULL.

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found.

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled.

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found.

NSM_DCB_API_SET_ERR_INTERFACE_NO_ETS when ETS is not enabled on the interface.

nsm_dcb_show_app_by_bridge

This function displays application priority information configured on a given bridge.

Syntax

```
s_int32_t
nsm_dcb_show_app_by_bridge (struct cli *cli, char *bridge_name)
```

Input Parameters

`cli`	CLI structure
`bridge_name`	Bridge ID

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds.

HAL Messages

The following are the HAL messages sent to HSL.

- `HAL_MSG_DCB_INIT`
- `HAL_MSG_DCB_DEINIT`
- `HAL_MSG_DCB_ENABLE`
- `HAL_MSG_DCB_DISABLE`
- `HAL_MSG_DCB_ETS_ENABLE`
- `HAL_MSG_DCB_ETS_DISABLE`
- `HAL_MSG_DCB_IF_ENABLE`
- `HAL_MSG_DCB_IF_DISABLE`
- `HAL_MSG_DCB_ETS_IF_ENABLE`
- `HAL_MSG_DCB_ETS_IF_DISABLE`
- `HAL_MSG_DCB_ETS_SELECT_MODE`
- `HAL_MSG_DCB_ETS_ADD_PRI_TO_TCG`
- `HAL_MSG_DCB_ETS_REMOVE_PRI_FROM_TCG`
- `HAL_MSG_DCB_ETS_TCG_BW_SET`

CHAPTER 4 DCB-PFC Functions

This chapter contains the APIs for DCB-PFC (Priority-based Flow Control). It includes the following APIs:

Function	Description
hal_dcb_get_pfc_stats	Sends a HAL_MSG_DCB_PFC_STATS message to HSL.
hal_dcb_pfc_bridge_enable	Sends a HAL_MSG_DCB_PFC_ENABLE message to HSL to enable DCB-PFC.
hal_dcb_pfc_bridge_disable	Sends a HAL_MSG_DCB_PFC_DISABLE message to HSL to disable DCB-PFC.
hal_dcb_pfc_interface_enable	Sends a HAL_MSG_DCB_PFC_IF_ENABLE message to HSL to enable DCB-PFC.
hal_dcb_pfc_interface_disable	Sends a HAL_MSG_DCB_PFC_IF_DISABLE message to HSL to disable DCB-PFC.
hal_dcb_enable_pfc_priority	Sends HAL_MSG_DCB_PFC_ADD_PRI message to HSL to enable DCB-PFC.
hal_dcb_disable_pfc_priority	Sends HAL_MSG_DCB_PFC_DEL_PRI message to HSL to disable DCB-PFC.
hal_dcb_set_pfc_cap	Sends a HAL_MSG_DCB_PFC_CAP_SET message to HSL.
hal_dcb_select_pfc_mode	Send a HAL_MSG_DCB_PFC_SELECT_MODE message to HSL
hal_dcb_set_pfc_lda	Sends a HAL_MSG_DCB_PFC_LINK_DELAY_ALLOWANCE message to HSL.
nsm_dcb_init_interface	Initializes the traffic class group settings on the interface to default values.
nsm_dcb_deinit_interface	De-initializes the traffic class group settings from an interface.
nsm_dcb_pfc_bridge_disable	Disables DCB-PFC from the switch level.
nsm_dcb_pfc_bridge_enable	Enables DCB-PFC at the switch level.
nsm_dcb_pfc_disable_interface	Disables DCB-PFC from an interface.
nsm_dcb_pfc_enable_interface	Enables PFC on an interface.
nsm_dcb_pfc_set_advertise_interface	Enables the “advertise” setting for PFC on an interface.
nsm_dcb_pfc_set_willing_interface	Enables the “willing” setting for PFC on an interface.
nsm_dcb_add_pfc_priority	Enables PFC on priorities for an interface.
nsm_dcb_remove_pfc_priority	Disables PFC on priorities from an interface.
nsm_dcb_set_pfc_cap	Sets the maximum number of priorities that can be enabled on an interface with PFC.
nsm_dcb_set_pfc_lda	Sets the PFC link delay allowance on an interface.
nsm_dcb_show_pfc_details_by_bridge	Displays PFC details for a given bridge.

Function	Description
nsm_dcb_show_pfc_details_by_interface	Displays PFC details for a given interface.
nsm_dcb_show_pfc_stats_by_bridge	Displays statistics for PFC requests sent on a given bridge.
nsm_dcb_show_pfc_stats_by_interface	Displays statistics for PFC requests sent on a given interface.
check_interface_pfc_state	Checks the PFC state of an interface.
nsm_dcb_update_pfc_xchg_config	Allows DCBX to apply configuration on to an operating configuration.
nsm_dcb_apply_pfc_config	Updates DCBX for the exchanged parameter and updates the configuration in the hardware directly for local configuration.
nsm_dcb_avl_traverse_disable_pfc	Disables the PFC DCB interface node information.

hal_dcb_get_pfc_stats

This function sends a `HAL_MSG_DCB_PFC_STATS` message to HSL.

```
s_int32_t
hal_dcb_get_pfc_stats (char *bridge_name, s_int32_t ifindex,
                       ut_int64_t *pause_sent ,ut_int64_t *pause_rcvd
```

Input Parameters

`bridge_name`	Bridge ID
`ifindex`	Interface index

Output Parameters

`pause_sent`	Pauses sent
`pause_rcvd`	Pauses received

Return Values

-1

hal_dcb_pfc_bridge_enable

This function sends a `HAL_MSG_DCB_PFC_ENABLE` message to HSL to enable DCB-PFC in the hardware at the switch level. The message passes the default configuration parameter to HSL.

Syntax

```
s_int32_t hal_dcb_pfc_bridge_enable (char *bridge_name)
```

Input parameter

`bridge_name`	Bridge name

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_pfc_bridge_disable

This HAL API sends a `HAL_MSG_DCB_PFC_DISABLE` message to HSL to disable DCB-PFC in the hardware at the switch level.

Syntax

```
s_int32_t hal_dcb_pfc_bridge_disable (char *bridge_name)
```

Input parameter

`bridge_name`	Bridge name

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_pfc_interface_enable

This function sends a `HAL_MSG_DCB_PFC_IF_ENABLE` message to HSL to enable DCB-PFC in the hardware on the interface.

Syntax

```
s_int32_t
hal_dcb_pfc_interface_enable (char *bridge_name, s_int32_t ifindex)
```

Input Parameters

`bridge_name`	Bridge ID
`ifindex`	Interface index

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_pfc_interface_disable

This function sends a HAL_MSG_DCB_PFC_IF_DISABLE message to HSL to disable DCB-PFC in the hardware on the interface.

Syntax

```
s_int32_t
hal_dcb_pfc_interface_disable (char *bridge_name, s_int32_t ifindex)
```

Input Parameters

`bridge_name`	Bridge ID
`ifindex`	Interface index

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_enable_pfc_priority

This HAL API sends a `HAL_MSG_DCB_PFC_ADD_PRI` message to HSL to enable DCB-PFC in the hardware on a given interface. The message passes the Interface index, priority, and status.

Syntax

```
s_int32_t hal_dcb_enable_pfc_priority (char *bridge_name, s_int32_t ifindex, u_int8_t
priority, bool_t status)
```

Input Parameters

`ifindex`	Interface index
`bridge_name`	Bridge name

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_disable_pfc_priority

This HAL API sends a `HAL_MSG_DCB_PFC_DEL_PRI` message to HSL to disable DCB-PFC in the hardware on a given interface.

Syntax

```
s_int32_t
hal_dcb_disable_pfc_priority (char *bridge_name, s_int32_t ifindex, s_int8_t pri)
```

Input Parameters

`bridge_name`	Bridge name
`ifindex`	Interface index
`pri`	Priority

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_set_pfc_cap

This HAL API sends a `HAL_MSG_DCB_PFC_CAP_SET` message to HSL, so that HSL can set the PFC cap value in the PFC configuration table. The message passes the Interface index and cap value.

Syntax

```
s_int32_t hal_dcb_set_pfc_cap (char *bridge_name, s_int32_t ifindex, u_int8_t cap)
```

Input Parameters

`bridge_name`	Bridge name.
`cap`	PFC capability value.
`ifindex`	Interface index.

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_select_pfc_mode

This HAL API send a `HAL_MSG_DCB_PFC_SELECT_MODE` message to HSL, so that HSL can sets or unsets the willing bit in the PFC configuration table based on the mode. The message passes the Interface index and mode value.

Syntax

```
s_int32_t
hal_dcb_select_pfc_mode (char *bridge_name, s_int32_t ifindex,
                         enum hal_dcb_mode mode)
```

Input Parameters

`bridge_name`	Bridge name
`ifindex`	Interface index
`mode`	Mode

Output Parameters

None

Return

HAL_SUCCESS

hal_dcb_set_pfc_lda

This function sends a `HAL_MSG_DCB_PFC_LINK_DELAY_ALLOWANCE` message to HSL, so that HSL can set the link delay allowance in the PFC configuration table. The message passes the Interface index and allowance value.

Syntax

```
s_int32_t
hal_dcb_set_pfc_lda (char *bridge_name, s_int32_t ifindex, u_int32_t lda)
```

Input Parameters

`bridge_name`	Bridge name.
`ifindex`	Interface index.
`lda`	Allowance value

Output Parameters

None

Return

HAL_SUCCESS

nsm_dcb_init_interface

This function initializes the traffic class group settings on the interface to default values.

Syntax

```
struct nsm_dcb_if *
nsm_dcb_init_interface (struct interface *ifp)
```

Input Parameters

`ifp`	Interface pointer

Output Parameters

None

Return Values

NULL

nsm_dcb_deinit_interface

This function de-initializes the traffic class group settings on the interface to default values.

Syntax

```
s_int32_t
nsm_dcb_deinit_interface (struct interface *ifp)
```

Input Parameters

`ifp`	Interface pointer

Output Parameters

None

Return Values

NULL

nsm_dcb_pfc_bridge_disable

This function disables DCB-PFC from the switch level.

Syntax

```
s_int32_t
nsm_dcb_pfc_bridge_disable (u_int32_t vr_id, char *bridge_name)
```

Input Parameters

`vr_id`	Virtual Router ID.
`bridge_name`	Switch name on which DCB should be initiate.

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS

NSM_DCB_API_ERR_DISABLE_PFC_INTERFACE

nsm_dcb_pfc_bridge_enable

This function enables DCB-PFC at the switch level.

Syntax

```
s_int32_t
nsm_dcb_pfc_bridge_enable (u_int32_t vr_id, char *bridge_name)
```

Input Parameters

`vr_id`	Virtual Router ID.
`bridge_name`	Switch name on which DCB should be initiate.

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS

nsm_dcb_pfc_disable_interface

This function disables DCB-PFC on an interface.

Syntax

```
s_int32_t
nsm_dcb_pfc_disable_interface (struct interface *ifp)
```

Input Parameters

`ifp`	Interface pointer

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found

NSM_DCB_API_SET_ERR_NO_PFC when PFC is not enabled

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found

NSM_DCB_API_SET_ERR_INTERFACE_NO_PFC when PFC is not enabled

nsm_dcb_pfc_enable_interface

This function enables DCB-PFC on an interface.

Syntax

```
s_int32_t
nsm_dcb_pfc_enable_interface (struct interface *ifp, enum nsm_dcb_mode mode)
```

Input Parameters

`ifp`	Interface pointer
`mode`	One of these values from `nsm/L2/nsm_dcb.h`:

`NSM_DCB_MODE_ON`

`NSM_DCB_MODE_AUTO`

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_DCB_ERR_AGG_PORT when the interface is an aggregator

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found

NSM_DCB_API_SET_ERR_NO_PFC when PFC is not enabled

NSM_DCB_API_ERR_LINK_FLOW_CTRL_ENABLE when flow control is enabled on the interface

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found

NSM_DCB_API_SET_ERR_INTERFACE_PFC_EXISTS when PFC is already enabled

NSM_DCB_API_SET_ERR_NO_MEM when there is an error allocating memory

nsm_dcb_pfc_set_advertise_interface

This function enables the “advertise” setting for PFC on an interface.

Syntax

```
s_int32_t
nsm_dcb_pfc_set_advertise_interface (struct interface *ifp,
                                     bool_t advertise)
```

Input Parameters

`ifp`	Interface pointer
`advertise`	Whether advertise is enabled is disabled.

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled

NSM_DCB_API_SET_ERR_NO_PFC when PFC is not enabled

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found

NSM_DCB_API_SET_ERR_INTERFACE_NO_PFC when PFC is not enabled on the interface

NSM_DCB_API_SET_ERR_INTERFACE_NO_ETS when ETS is not enabled on the interface

nsm_dcb_pfc_set_willing_interface

This function enables the “willing” setting for PFC on an interface.

Syntax

```
s_int32_t
nsm_dcb_pfc_set_willing_interface (struct interface *ifp,
                                   bool_t willing)
```

Input Parameters

`ifp`	Interface pointer
`willing`	Whether willing is enabled is disabled.

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled

NSM_DCB_API_SET_ERR_NO_PFC when PFC is not enabled

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found

NSM_DCB_API_SET_ERR_INTERFACE_NO_PFC when PFC is not enabled on the interface

nsm_dcb_add_pfc_priority

This function enables PFC on priorities for an interface.

Syntax

```
s_int32_t
nsm_dcb_add_pfc_priority (struct interface *ifp, u_int8_t priority_map)
```

Input Parameters

`ifp`	Interface pointer
`priority_map`	Bitmap of priorities to enable

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled

NSM_DCB_API_SET_ERR_NO_PFC when PFC is not enabled

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found

NSM_DCB_API_SET_ERR_INTERFACE_NO_PFC when PFC is not enabled on the interface

NSM_DCB_API_SET_ERR_EXCEED_PFC_CAP when the priorities being enabled exceeds the maximum number of priorities

nsm_dcb_remove_pfc_priority

This function disables PFC on priorities for an interface.

Syntax

```
s_int32_t
nsm_dcb_remove_pfc_priority (struct interface *ifp, u_int8_t priority_map)
```

Input Parameters

`ifp`	Interface pointer
`priority_map`	Bitmap of priorities to disable

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled

NSM_DCB_API_SET_ERR_NO_PFC when PFC is not enabled

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found

NSM_DCB_API_SET_ERR_INTERFACE_NO_PFC when PFC is not enabled on the interface

NSM_DCB_API_SET_WRONG_PRIORITY_MAP when one or more priorities to disable are not in the priority map

nsm_dcb_set_pfc_cap

This function sets the maximum number of priorities that can be enabled on an interface with PFC.

Syntax

```
s_int32_t
nsm_dcb_set_pfc_cap(struct interface *ifp, u_int8_t cap)
```

Input Parameters

`ifp`	Interface pointer
`cap`	Maximum number of priorities on which PFC can be enabled

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled

NSM_DCB_API_SET_ERR_NO_PFC when PFC is not enabled

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found

NSM_DCB_API_SET_ERR_INTERFACE_NO_PFC when PFC is not enabled on the interface

NSM_DCB_API_SET_ERR_PFC_CAP when the maximum number of priorities being set exceeds current number of the priorities

nsm_dcb_set_pfc_lda

This function sets the PFC link delay allowance on an interface.

Syntax

```
s_int32_t
nsm_dcb_set_pfc_lda(struct interface *ifp, u_int32_t lda)
```

Input Parameters

`ifp`	Interface pointer
`lda`	Link delay allowance

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled

NSM_DCB_API_SET_ERR_NO_PFC when PFC is not enabled

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found

NSM_DCB_API_SET_ERR_INTERFACE_NO_PFC when PFC is not enabled on the interface

NSM_DCB_API_SET_ERR_INTERFACE_NO_ETS when ETS is not enabled on the interface

NSM_DCB_API_SET_ERR_HW_NO_SUPPORT when the link delay allowance cannot be set in the hardware

nsm_dcb_show_pfc_details_by_bridge

This function displays PFC details for a given bridge.

Syntax

```
s_int32_t
nsm_dcb_show_pfc_details_by_bridge (struct cli *cli,
                                    char *bridge_name)
```

Input Parameters

`cli`	CLI structure
`bridge_name`	Bridge ID

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found

NSM_DCB_API_SET_ERR_NO_PFC when PFC is not enabled

nsm_dcb_show_pfc_details_by_interface

This function displays PFC details for a given interface.

Syntax

```
s_int32_t
nsm_dcb_show_pfc_details_by_interface (struct cli* cli,
                                       struct interface *ifp)
```

Input Parameters

`cli`	CLI structure
`ifp`	Interface pointer

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_DCB_API_SET_ERR_INTERFACE if the interface is NULL

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found

NSM_DCB_API_SET_ERR_NO_PFC when PFC is not enabled

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found

NSM_DCB_API_SET_ERR_INTERFACE_NO_PFC when PFC is not enabled on the interface

nsm_dcb_show_pfc_stats_by_bridge

This function displays statistics for PFC requests sent on a given bridge.

Syntax

```
s_int32_t
nsm_dcb_show_pfc_stats_by_bridge (struct cli *cli, char *bridge_name)
```

Input Parameters

`cli`	CLI structure
`bridge_name`	Bridge ID

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found

NSM_DCB_API_SET_ERR_NO_PFC when PFC is not enabled

nsm_dcb_show_pfc_stats_by_interface

This function displays statistics for PFC requests sent on a given interface.

Syntax

```
s_int32_t
nsm_dcb_show_pfc_stats_by_interface (struct cli* cli, struct interface *ifp)
```

Input Parameters

`cli`	CLI structure
`ifp`	Interface pointer

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_DCB_API_SET_ERR_INTERFACE if the interface is NULL

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found

NSM_DCB_API_SET_ERR_NO_PFC when PFC is not enabled

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found

NSM_DCB_API_SET_ERR_INTERFACE_NO_PFC when PFC is not enabled on the interface

NSM_DCB_API_SET_ERR_HW_NO_SUPPORT when the hardware does not support PFC statistics

check_interface_pfc_state

This function checks the PFC state of an interface.

Syntax

```
bool_t
check_interface_pfc_state(struct interface *ifp)
```

Input Parameters

`ifp`	Interface pointer

Output Parameters

None

Return Values

PAL_TRUE when the function succeeds

PAL_FALSE when the function fails

nsm_dcb_update_pfc_xchg_config

This function allows DCBX to apply configuration on to an operating configuration.

Syntax

```
s_int32_t
nsm_dcb_update_pfc_xchg_config (struct interface *ifp,
                                struct pfc_config *pfc_config)
```

Input Parameters

`ifp`	Interface pointer
`pfc_config`	Pointer to the PFC configuration.

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_DCB_API_SET_ERR_DCB_INTERFACE

NSM_DCB_API_SET_ERR_HW_NO_SUPPORT

nsm_dcb_apply_pfc_config

This function updates DCBX for the exchanged parameter and updates the configuration in the hardware directly for local configuration.

Syntax

```
s_int16_t
nsm_dcb_apply_pfc_config (struct interface *ifp,
                    struct pfc_config *pfc_config)
```

Input Parameters

`ifp`	Interface pointer
`pfc_config`	Pointer to the PFC configuration.

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_DCB_API_SET_ERR_DCB_INTERFACE

NSM_DCB_API_SET_ERR_HW_NO_SUPPORT

nsm_dcb_avl_traverse_disable_pfc

This function disables the PFC DCB interface node information.

Syntax

```
s_int32_t
nsm_dcb_avl_traverse_disable_pfc (void_t *node_info, void_t *arg1)
```

Input Parameters

`node_info`	Node information
`arg1`	Pointer to the first argument.

Output Parameters

None

Return Values

NULL

HAL Messages

The following list the HAL messages that are sent with HSL.

- HAL_MSG_DCB_PFC_DISABLE
- HAL_MSG_DCB_PFC_ENABLE
- HAL_MSG_DCB_PFC_IF_ENABLE
- HAL_MSG_DCB_PFC_SELECT_MODE

CHAPTER 5 DCB-QCN Functions

This chapter contains the APIs for DCB-QCN (Quantized Congestion Notification). It includes the following APIs:

Function	Description
hal_dcb_qcn_add_cnpv	Sends a HAL_MSG_QCN_ADD_CNPV message to HSL.
hal_dcb_qcn_remove_cnpv	Sends a HAL_MSG_QCN_REMOVE_CNPV message to HSL.
hal_dcb_qcn_cp_enable	Sends a HAL_MSG_QCN_CP_ENABLE message to HSL.
hal_dcb_qcn_cp_disable	Sends a HAL_MSG_QCN_CP_DISABLE message to HSL.
hal_dcb_qcn_get_config	Sends a HAL_MSG_DCB_QCN_GET_CONFIG message to HSL
hal_dcb_qcn_get_config_cp	Sends a HAL message to HSL, passing the nsm_hal_cp_data
hal_dcb_qcn_init	Sends a HAL_MSG_DCB_QCN_INIT message to HSL to initialize QCN in the hardware.
hal_dcb_qcn_deinit	Sends a HAL_MSG_DCB_QCN_DEINIT message to HSL to deinitialize QCN in the hardware.
hal_dcb_qcn_set_cnm_priority	Sends a HAL_MSG_DCB_QCN_CNM_PRI message to HSL
hal_dcb_qcn_set_defense_mode	Sends a HAL_MSG_QCN_SET_DEFENSE_MODE message to HSL to set the defense mode
nsm_dcb_create_cp	Creates a Congestion Point (CP) for CNPV on an interface.
nsm_qcn_disable	Disables QCN on a switch.
nsm_qcn_enable	Enables QCN on a switch.
nsm_dcb_qcn_priority_disable	Disables a CNPV on a switch.
nsm_dcb_qcn_priority_enable	Enables a CNPV on a switch.
nsm_dcb_remove_cp	Removes a Congestion Point (CP) for CNPV on an interface
nsm_dcb_set_qcn_mode_global	Sets the defense mode of CNPV on a switch (bridge).

hal_dcb_qcn_add_cnpv

This function sends a HAL_MSG_QCN_ADD_CNPV message to HSL to enable a Congestion Notification Priority Value (CNPV) in the hardware.

Syntax

```
s_int32_t
hal_dcb_qcn_add_cnpv (char *bridge_name, s_int8_t cnpv, u_int8_t alternate_priority)
```

Input Parameters

`bridge_name`	Bridge ID
`cnpv`	CNPV
`alternate_priority`	Alternate priority

Output Parameters

None

Return

-1

0

hal_dcb_qcn_remove_cnpv

This function sends a `HAL_MSG_QCN_REMOVE_CNPV` message to HSL to disable a Congestion Notification Priority Value (CNPV) in the hardware.

Syntax

```
s_int32_t
hal_dcb_qcn_remove_cnpv (char *bridge_name, s_int8_t cnpv)
```

Input Parameters

`bridge_name`	Bridge ID
`cnpv`	CNPV

Output Parameters

None

Return

-1

0

hal_dcb_qcn_cp_enable

This function sends a `HAL_MSG_QCN_CP_ENABLE` message to HSL to enable the interface to act as a congestion point (CP) in the hardware. After calling this function, the hardware sends Congestion Notification Messages (CNMs) when it detects congestion. If the value of any of the parameters (except `ifindex`) is 0, then the hardware sets the default value.

Syntax

```
s_int32_t
hal_dcb_qcn_cp_enable (s_int32_t ifindex, u_int8_t cnpv, u_int32_t sample_base,
                       float weight, u_int32_t min_hdr_octects)
```

Input Parameters

`ifindex`	Interface index
`cnpv`	Congestion Notification Priority Value (CNPV)
`sample_base`	The minimum number of octets to enqueue in the CP's queue between transmissions of CNMs <10000-4294967295>. The default is 150,000.
`weight`	The weight change in queue length used to calculate whether the queue length is moving toward or away from the target number of octets for the CP's queue <-10 - 10>. The default is one (1).
`min_hdr_octet`	The minimum number of octets that the CP returns in the MSDU (MAC Service Data Unit) field of each CNM it generates <0-64>. The default is zero (0).

Output Parameters

None

Return

-1

0

hal_dcb_qcn_cp_disable

This function sends a `HAL_MSG_QCN_CP_DISABLE` message to HSL to disable the Congestion Notification Priority Value (CNPV) at the given interface in the hardware.

Syntax

```
s_int32_t
hal_dcb_qcn_cp_disable (s_int32_t ifindex, u_int8_t cnpv)
```

Input Parameters

`ifindex`	Interface index
`cnpv`	CNPV

Output Parameters

None

Return

-1

0

hal_dcb_qcn_get_config

This function sends a `HAL_MSG_DCB_QCN_GET_CONFIG` message to HSL, passing the `nsm_hal_qcn_data` structure which HSL fills.

Syntax

```
s_int32_t
hal_dcb_qcn_get_config (char *bridge_name, struct nsm_hal_qcn_data *data)
```

Input Parameters

`bridge_name`	Bridge ID
`data`	A pointer to an instance of the `nsm_hal_qcn_data` struct defined in `nsm/L2/nsm_dcb.h`

Output Parameters

None

Return

-1

0

hal_dcb_qcn_get_config_cp

This function sends a HAL message to HSL, passing the nsm_hal_cp_data structure which HSL fills up with CP details and returns back.

Syntax

```
s_int32_t
hal_dcb_qcn_get_config_cp_cpid (char *bridge_name,
                      struct nsm_hal_cp_data *data, u_int32_t cp_id)
```

Input Parameters

`bridge_name`	Bridge ID
`data`	A pointer to an instance of the `nsm_hal_cp_if_data` struct defined in `nsm/L2/nsm_dcb.h`
`cp_id`	

Output Parameters

None

Return

-1

0

hal_dcb_qcn_init

This function sends a `HAL_MSG_DCB_QCN_INIT` message to HSL to initialize QCN in the hardware. This function only enables the hardware to recognize the Congestion Notification Tag (CN-Tag) and to set the Congestion Notification Domain (CND). This function does not start sending Congestion Notification Messages (CNMs).

Syntax

```
s_int32_t
hal_dcb_qcn_init (char *bridge_name, u_int8_t transmit_priority)
```

Input Parameters

`bridge_name` Bridge ID

`transmit_priority`
Priority used for congestion notification messages

Output Parameters

None

Return

-1

0

hal_dcb_qcn_deinit

This function sends a `HAL_MSG_DCB_QCN_DEINIT` message to HSL to deinitialize QCN in the hardware.

Syntax

```
s_int32_t
hal_dcb_qcn_deinit (char *bridge_name)
```

Input Parameters

`bridge_name` Bridge ID

Output Parameters

None

Return

-1

0

hal_dcb_qcn_set_cnm_priority

This function sends a `HAL_MSG_DCB_QCN_CNM_PRI` message to HSL so that the hardware will use the correct priority in Congestion Notification Messages (CNMs).

Syntax

```
s_int32_t
hal_dcb_qcn_set_cnm_priority (char *bridge_name, u_int8_t priority)
```

Input Parameters

`bridge_name` Bridge ID

`priority` Priority

Output Parameters

None

Return

-1

0

hal_dcb_qcn_set_defense_mode

This function sends a `HAL_MSG_QCN_SET_DEFENSE_MODE` message to HSL to set the defense mode and the alternate priority in the hardware on the interface.

Syntax

```
s_int32_t
hal_dcb_qcn_set_defense_mode (s_int32_t ifindex, u_int8_t cnpv,
                              u_int32_t defense_mode, u_int32_t alt_priority)
```

Input Parameters

`ifindex`	Interface index
`cnpv`	CNPV
`defense_mode`	Defense mode
`alt_priority`	Alternate priority

Output Parameters

None

Return

-1

0

nsm_dcb_create_cp

This function creates a Congestion Point (CP) for a Congestion Notification Priority Value (CNPV) on an interface.

Syntax

```
s_int32_t
nsm_dcb_create_cp (u_int32_t vr_id, char *ifname, u_int8_t cnpv,
                   u_int32_t sample_base, s_int32_t weight,
                   u_int32_t min_hdr_octet, u_int8_t flag)
```

Input Parameters

`vr_id`	Virtual router ID
`ifname`	Interface name
`cnpv`	CNPV.

Note: If you do not set the `cnpv` bit, then the values for the other bits that you do set are applied to all CNPVs. The default value is used for any other bit that you do not set.

`sample_base`	The minimum number of octets to enqueue in the CP's queue between transmissions of Congestion Notification Messages (CNMs) <10000-4294967295>. The default is 150,000.

`weight`	The weight change in queue length used to calculate whether the queue length is moving toward or away from the target number of octets for the CP's queue <-10 - 10>. The default is one (1).
`min_hdr_octet`	The minimum number of octets that the CP returns in the MSDU (MAC Service Data Unit) field of each CNM it generates <0-64>. The default is zero (0).
`flag`	Specify which values you are supplying in the `cnpv`, `sample_base`, `weight`, and `min_hdr_octet` parameters by setting bits in this parameter.

Output Parameters

None

Return

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found

NSM_DCB_API_SET_ERR_INTERFACE when the interface is not found

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled

NSM_DCB_API_SET_ERR_NO_QCN when QCN is not enabled

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found

NSM_DCB_API_SET_ERR_NO_MEM when there is an error allocating memory

NSM_DCB_API_SET_ERR_HW when a CP cannot be enabled in the hardware

NSM_DCB_API_SET_ERR_NO_CNPV when a CP is not configured for a given CNPV on the interface

NSM_DCB_API_SET_ERR_NO_CNPV_CONF when a given CNPV is not configured

nsm_qcn_disable

This function disables QCN on a switch (bridge).

Syntax

```
s_int32_t nsm_qcn_disable (u_int32_t vr_id, char *bridge_name)
```

Input Parameters

`vr_id`	Virtual router ID
`bridge_name`	Bridge ID

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_QCN when QCN is not enabled

NSM_DCB_API_SET_ERR_HW when QCN cannot be disabled in the hardware

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled

nsm_qcn_enable

This function enables QCN on a switch (bridge).

Syntax

```
s_int32_t
nsm_qcn_enable (u_int32_t vr_id, char *bridge_name, u_int8_t transmit_priority)
```

Input Parameters

`vr_id`	Virtual router ID
`bridge_name`	Bridge ID
`transmit_priority`	Priority used for congestion notification messages

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_VLAN_ERR_BRIDGE_NOT_VLAN_AWARE when the bridge is not VLAN-aware

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled

NSM_DCB_API_SET_ERR_NO_MEM when there is an error allocating memory

NSM_DCB_API_SET_ERR_HW when there is an error enabling QCN in the hardware

nsm_dcb_qcn_priority_disable

This function disables a Congestion Notification Priority Value (CNPV) on a switch (bridge).

Syntax

```
s_int32_t
nsm_dcb_qcn_priority_disable (u_int32_t vr_id, char *bridge_name,
                              u_int8_t cnpv)
```

Input Parameters

`vr_id`	Virtual router ID
`bridge_name`	Bridge ID
`cnpv`	CNPV to disable

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled

NSM_DCB_API_SET_ERR_NO_QCN when QCN is not enabled

NSM_DCB_API_SET_ERR_NO_CNPV when a CP is not configured for a given CNPV on the interface

NSM_DCB_API_SET_ERR_HW when there is an error disabling a CNPV in the hardware

nsm_dcb_qcn_priority_enable

This function enables a Congestion Notification Priority Value (CNPV) on a switch (bridge). A CNPV corresponds to a class of applications or a single application, such as interprocess communications or disk storage, that have different requirements for network resources as such as latency or bandwidth. There are total of eight CNPVs with one reserved for “best-effort” traffic, meaning you can assign up to seven CNPVs.

Syntax

```
s_int32_t
nsm_dcb_qcn_priority_enable (u_int32_t vr_id, char *bridge_name,
                             u_int8_t priority)
```

Input Parameters

`vr_id`	Virtual router ID
`bridge_name`	Bridge ID
`priority`	CNPV to enable

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled

NSM_DCB_API_SET_ERR_NO_QCN when QCN is not enabled

NSM_DCB_API_SET_ERR_QCN_ALT_PRIORITY when there is an error calculating an alternate priority

NSM_DCB_API_SET_ERR_HW when a CNPV cannot be enabled in the hardware

nsm_dcb_remove_cp

This function removes a Congestion Point (CP) for a Congestion Notification Priority Value (CNPV) from an interface.

Syntax

```
s_int32_t
nsm_dcb_remove_cp (u_int32_t vr_id, char *ifname, u_int8_t cnpv)
```

Input Parameters

`vr_id`	Virtual router ID
`ifname`	Interface name
`cnpv`	CNPV

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found

NSM_DCB_API_SET_ERR_INTERFACE when the interface is not found

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled

NSM_DCB_API_SET_ERR_NO_QCN when QCN is not enabled

NSM_DCB_API_SET_ERR_DCB_INTERFACE when the DCB interface is not found

NSM_DCB_API_SET_ERR_HW when a CP cannot be disabled in the hardware

nsm_dcb_set_qcn_mode_global

This function sets the defense mode of a Congestion Notification Priority Value (CNPV) on a switch (bridge).

Syntax

```
s_int32_t
nsm_dcb_set_qcn_mode_global (u_int32_t vr_id, char *bridge_name, u_int8_t cnpv,
                             enum nsm_dcb_qcn_mode mode,
                             enum nsm_dcb_qcn_defense_mode defense_mode,
                             u_int8_t alternate_priority)
```

Input Parameters

`vr_id`	Virtual router ID
`bridge_name`	Bridge ID
`cnpv`	CNPV
`mode`	One of these constants from the `nsm_dcb_qcn_mode` enum in `nsm/L2/nsm_dcb.h`:

`NSM_DCB_QCN_MODE_ADMIN`

Explicitly set the defense mode to the value specified in the `defense_mode` parameter

NSM_DCB_QCN_MODE_AUTO

Set the defense mode according to LLDP Congestion Notification TLVs and set the alternate priority to the cncpAutoAltPri variable as defined in the 802.1Qau standard.

`defense_mode` One of these constants from the `nsm_dcb_qcn_defense_mode` enum in `nsm/L2/nsm_dcb.h`:

NSM_QCN_DEFENSE_MODE_DISABLE

Congestion notification is disabled for this CNPV.

NSM_QCN_DEFENSE_MODE_EDGE

The priority parameters of input frames are remapped to an alternate value. Congestion notification tags Congestion Notification Tags (CN-tags) are not output.

NSM_QCN_DEFENSE_MODE_INTERIOR

The priority parameters of input frames are not remapped to another value. CN-tags are not output.

NSM_QCN_DEFENSE_MODE_INTERIOR_READY

The priority parameters of input frames are not remapped to another value. CN-tags can be output.

`alternate_priority`

Alternate priority value

Output Parameters

None

Return Values

NSM_DCB_API_SET_SUCCESS when the function succeeds

NSM_DCB_API_SET_ERR_NO_NM when the NSM master is not found

NSM_BRIDGE_ERR_NOTFOUND when the bridge is not found

NSM_DCB_API_SET_ERR_NO_DCBG when the DCB bridge is not found or DCB or not enabled

NSM_DCB_API_SET_ERR_NO_QCN when QCN is not enabled

NSM_DCB_API_SET_ERR_NO_CNPV when a CP is not configured for a given CNPV on the interface

NSM_DCB_API_SET_ERR_HW when a defense mode cannot be set in the hardware

Index

S