# Intel® IXP400 Software: VLAN and QoS Application Version 2.0

*Programmer's Guide*

**October 2005**

Document Number: 301925, Revision: 002
October 2005

# Contents



## Figures

# Tables

# Revision History

| Date | Revision | Description |
|---|---|---|
| October 2005 | 002 | General updates. Replaced Section 6.0, “API Reference” on page 31. Change bars indicate areas of change. |
| September 2004 | 001 | Initial release. |

**This page is intentionally left blank.**

# 1.0 Introduction

## 1.1 What's New

The VLAN QoS API has been updated from version 1.0. This is documented in Section 6.0, "API Reference" on page 31.

**VLAN Features**

- Supports all VLAN groups (VLAN ID #1 to #4094) and all can be enabled simultaneously. Version 1.0 only supports 32 VLAN groups.
- Protocol-Based VLAN classification supports IPv6 and IPv4 protocol. Version 1.0 only supports IPv4.
- The following computing/processing tasks are off-loaded to NPE-level software:
  - — Acceptable Frame Type Checker
  - — VLAN Membership Filtering
  - — Frame Tagging and Tag-Removal on Egress

**QoS Features**

- Supports QoS on Ingress and Egress side. Version 1.0 only supports QoS on Ingress.
- Four traffic classes are supported for each side and for each NPE Ethernet port. Version 1.0 can support eight traffic classes on Ingress for each port.
- Automatic adjusting for queue length for traffic shapers is supported. The benefit is that the setting function for queue length, which while supported in Version 1.0, is no longer needed or supported.

## 1.2 Scope and Purpose

The purpose of this document is to provide high-level technical design information for the Intel® IXP400 Software VLAN and QoS Application v2.0. Based on Intel® IXP400 Software v2.0, the VLAN and QoS application is provided to implement IEEE 802.1Q VLAN (Virtual Local Area Networks), IEEE 802.1p User Priority to Traffic Class (TC) mappings, and Quality of Services (QoS) functionality for IPv4/IPv6 traffic using the IXP400 software.

This document covers the high-level functionality of the various modules, and describes their behavioral links. For a more complete understanding, you should review the API reference information provided in Section 6.0, "API Reference" on page 31, review the IxEthDB Functional Behavior section in the *Intel® IXP400 Software Programmer's Guide*, and review the VLAN and QoS Example Code user interface (as described in the *VLAN and QoS Application Version 2.0 Release Notes*), and the VLAN and QoS Example source code.

It is assumed that you are familiar with IEEE 802.1D Ethernet bridging and IEEE 802.1Q/p VLAN and Priority functionality.

## 1.3 Acronyms

| | |
|---|---|
| FIFO | First In, First Out |
| ID | Identification |
| IEEE | Institute of Electrical and Electronics Engineers |
| IO | Input / Output |
| IOCTL | I/O Control |
| LAN | Local Area Network |
| MAC | Media Access Controller |
| NPE | Network Processing Engine |
| OS | Operating System |
| PVID | Port VLAN ID |
| TC | Traffic Class |
| QoS | Quality of Service |
| VID | VLAN Identification |
| VLAN | Virtual LAN |

## 1.4 Related Documents

Additional Intel documents listed below are available from your field representative or from the following Web site:

http://www.intel.com/design/network/products/npfamily/docs/ixp4xx.htm

| Document Title | Document # |
|---|---|
| *Intel® IXP400 Software: VLAN and QoS Application Version 2.0 Release Notes* | N/A |
| *Intel® IXP400 Software Release 2.0 Software Release Notes* | N/A |
| *Intel® IXP400 Software Programmer's Guide* (for Release v2.0) | 252539-007 |
| *Intel® IXP400 Software Specification Update* | 273795 |
| IEEE Standards (IEEE Std 802.1D-1998) for Local Area and Metropolitan Networks, Media Access Control (MAC) Bridge | N/A |
| IEEE Standards (IEEE Std 802.1Q-1998) for Local Area and Metropolitan Networks, Virtual Bridged Local Area Networks | N/A |
| IEEE Standards (IEEE Std 802.1p-1998) for Traffic class expediting and dynamic multicast filtering | N/A |

# 2.0 Software Overview

## 2.1 Functionality Overview

### 2.1.1 VLAN Functionality

The VLAN (Virtual Local Area Networks) functionality behaves as described in the following example scenario:

1. A frame generated from one station is received by NPE Ethernet port 1.
2. The frame type (VLAN-tagged or VLAN-untagged) is evaluated for the acceptance test.

   If the frame is not acceptable by port 1, it is discarded; otherwise, the VLAN tag for the frame is determined. For a VLAN-tagged frame, the VLAN tag is obtained from the frame header; otherwise, it is determined by the VLAN tag of the ingress port (i.e., port 1) or by the classification rules. User Priority and VID (id of the VLAN which this frame should be grouped into) are included in the VLAN tag.

   In this example, the VID of the frame is 1.
3. The Ingress rule is then applied to determine if this frame should be discarded or kept.

   The Ingress rule determines whether the station is a member of the VLAN in which the frame is grouped.

   In our example, the frame has not been discarded and the bridging function component will decide which port this frame should be forwarding to.

4. Once the destination port (e.g., NPE Ethernet port 2) is decided, the egress filtering rule will be applied to make sure whether this frame can be transmitted on the destination port.

   Suppose the VLAN #1 is in the membership of port 2; therefore, the frame passes the filtering rule and is instructed by egress rule that its frame type as being submitted through port 2.

   To fit into the proper frame type, the frame will be tagged, un-tagged by egress port or be passed through.

### 2.1.2 QoS Functionality

Traffic can be classified into traffic classes. QoS Traffic Class Mapping maps each Ethernet frame into particular traffic classes according to the user priority field of the frame. When the IXP400-based system is connected to a VLAN machine, that machine is considered the ‘previous stat’ to the IXP400 system. The previous stat generates Ethernet frames with VLAN tags containing the user priority field.

The mapping table should be manageable per port to allow different QoS strategies on each individual port.

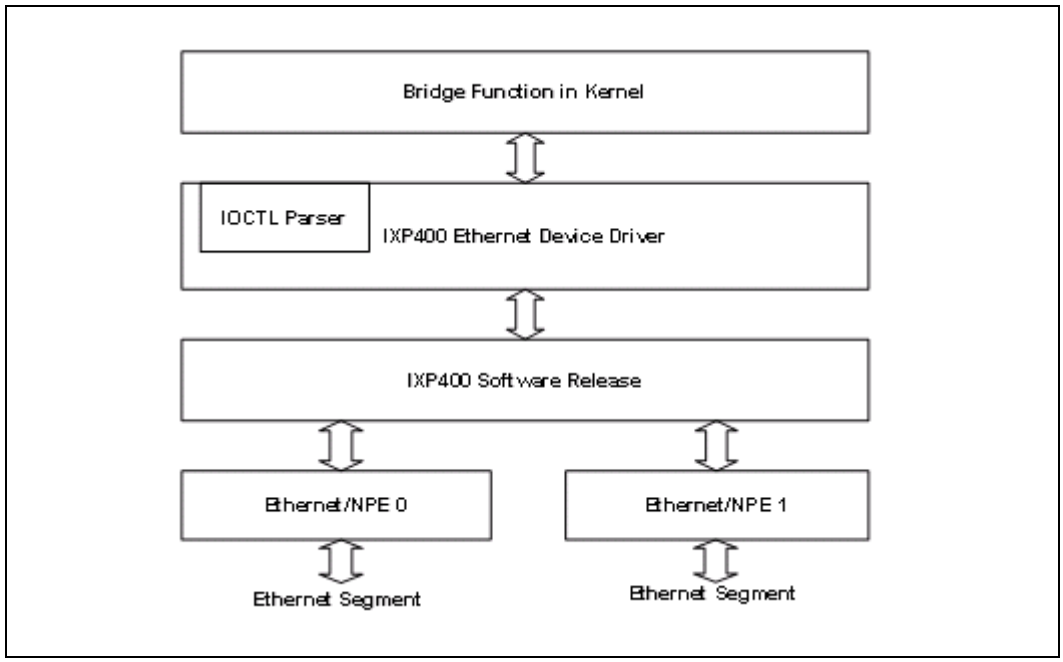
With the exception of bridging, ingress traffic is usually forwarded to the upper layers. This is usually data that needs further processing by the host processor, such as DSP (digital signal processing) applications or network routing. QoS can be applied to each traffic type differently according to their Traffic Class. For example, time-sensitive traffic (e.g., voice) is mapped into a traffic class different from time-insensitive traffic (e.g., routing frame). By setting higher priority for the voice traffic, the bandwidth for the voice data can be distributed to prevent transmission of the voice frames from being delayed/blocked by the routing frames. This would help ensure high voice quality in VoIP applications.

In the egress side of NPE Ethernet port, the QoS is also supported in the manner very similar to the ingress side. The egress traffics are classified and mapped into classes. Priority can be set to the individual classes. Careful assignment of priorities can help conserve transmission bandwidth for higher-priority traffic.

## 2.2 Software Architecture and High-Level Design

As depicted in Figure 1, the software architecture of Intel® IXP400 Software VLAN and QoS Application v2.0 is designed to integrate with the IXP400 software.

**Figure 1. Intel® IXP400 Software and Ethernet Device Driver Overview**




The Intel® IXP4XX Product Line of Network Processors contain Network Processing Engines (NPEs), which provide physical connectivity and processing of data to various interfaces. One function of the IXP400 software is to provide OS and upper-level applications access to these interfaces via a set of APIs. In the case of the VLAN and QoS Example Code, the two Ethernet NPE ports are the two physical links of an Ethernet bridge. The Ethernet device driver is the OS-specific code that provides access to these NPEs via the services of the IXP400 software.

The VLAN and QoS functionality is provided by a set of software modules that interface with the IXP400 Software, the OS-specific device driver for the NPE ports, and the OS-specific bridging software.

The modules do contain a minimal amount of OS-dependent code. When OS-specific code is used, it is enclosed by a compiler definition typically passed through to the makefiles from the IXP400 software build system.

**Figure 2. Software Architecture with the VLAN and QoS Example Code**

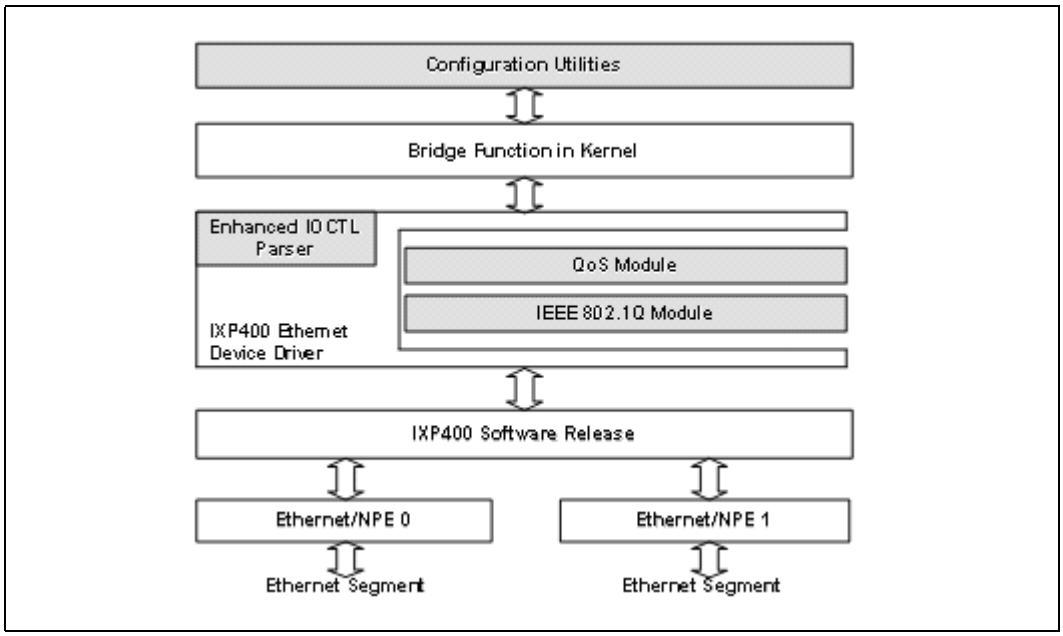


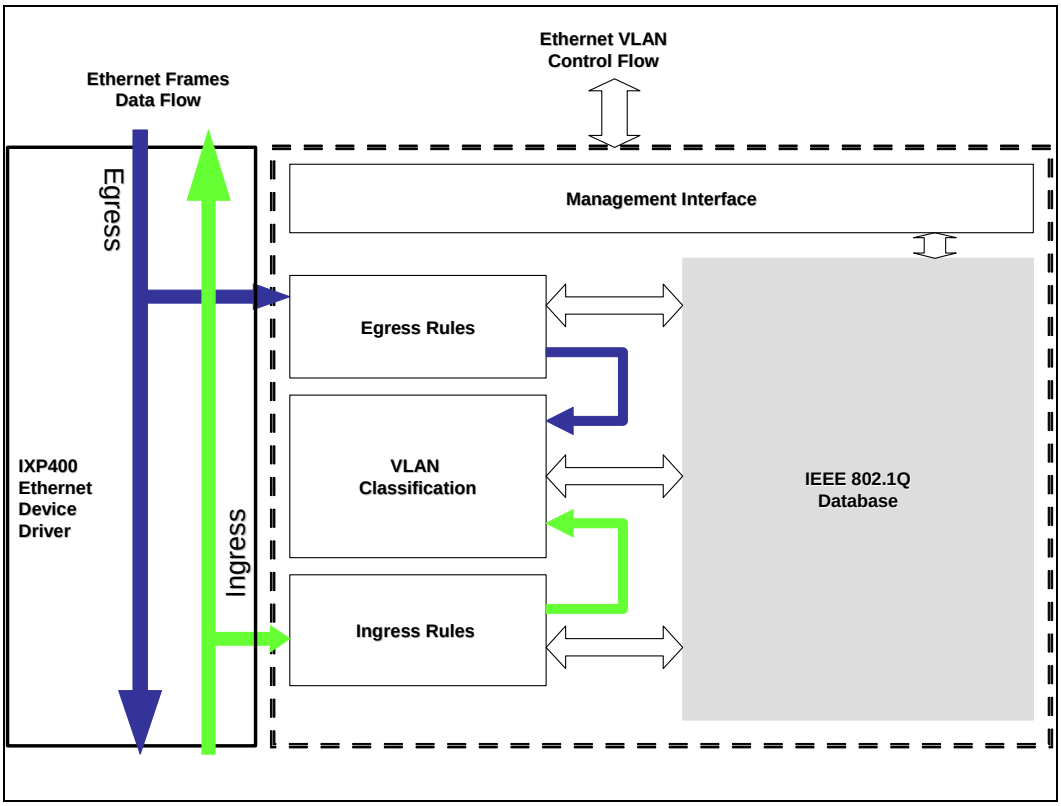

Figure 2 closely resembles Figure 1, but includes the VLAN and QoS Example Code. The two modules, one for 802.1Q VLAN and another for QoS processing, are inserted into the data path of the system. The Ethernet device driver uses these modules when VLAN-capable Ethernet frames are received or need to be sent.

The VLAN and QoS Example Code also provides control path capabilities. The Ethernet device driver's IOCTL parser is enhanced to recognize and execute the additional VLAN and QoS functionality.

# 3.0 802.1Q VLAN Module

This module implements the IEEE 802.1Q VLAN functionality. The module includes five software sub-components, which are briefly described below; later sections provide more sub-component detail. Ingress Rules, Egress Rules, and VLAN Classification deal with frame processing, the Database records all information supported by 802.1Q VLAN module, while the Management Interface deals with configuration for each component and provides public APIs for external modules. The general flow, shown in Figure 3, is described below.

**Figure 3. 802.1Q VLAN Module – Component View**




**LAN Module Sub-Components**

- Ingress Rules

  Two Ingress Rule filterings are supported: Acceptable Frame Types, and Ingress VLAN Membership. Since IXP400 Software v1.5, the filterings have been assisted by NPE firmware instead of by software running on the Intel XScale core. For Acceptable Frame Types filtering, the NPE determines if received frames are “VLAN-tagged” or “all frame types”. Frames are discarded if the reception port is not allowed to receive these types of frames. For Ingress VLAN Membership filtering, the frames are discarded if the VLAN group the frame carries is not in the membership table of the port from which the frame is received or transmitted through.

- VLAN Classification

  Determines VLAN Identification (VID) and User Priority of received frames (in ingress path) and transmission frames (in egress path), either by the criteria of Tag-Based, Port-Based, Protocol-Based, or MAC-Based VLAN.

- Egress Rules

  Two major features are supported: Egress VLAN Membership filtering and Rebuild Packet Header. The function of the Egress VLAN Membership filtering is the same as Ingress VLAN Membership filtering, except that it executes at the egress port. Rebuild Packet Header supports the ability to determine if transmission frames should be tagged or untagged, and then adds/removes/modifies the VLAN-tag header for the outbound frames.

- Management Interface

  Interface for maintaining database and public APIs.

- Databases

  Records all information and rules for the 802.1Q VLAN module. They include port-specific information, VLAN-group-specific information, and VLAN classification rules. The port-specific information is port-related such as PVID (Port VLAN Identification), Acceptable Frame Types parameter, and VLAN membership table. VLAN-specific information contains the Egress attributes (tagged or untagged). Classification rules include the MAC-based classification rules and Protocol (Layer 3/4)-based classification rules.

**Private Frame Buffer Memory**

Sixteen bytes of extra memory is reserved for each frame buffer used by the VLAN module for storing per-frame VID and user priority information, which is determined at the ingress port. This private memory is transparent to the IXP400 software.

**Ingress Path**

For inbound Ethernet Frames, the NPE firmware performs the Ingress rule for VLAN ingress processing. The Ingress Rules component analyzes the frame type, VLAN-tagged, Priority-tagged, or VLAN-untagged type (see Figure 4), of received frames and commences Acceptance Frame Type Check filtering. Frames are discarded if their frame types are not allowed on the reception port. If frames pass the Acceptance Frame Type Check filtering, the Ingress Rules component then calls the services of the VLAN Classification component to determine VLAN Identification (VID) and user priority of received frames. Next, the Database component is queried to get the (port) member set of the detected VLAN group and decide if the reception port is in the member set of that VLAN group or not. Frames are discarded if the reception port is not in the member set of the detected VLAN group. For frames that pass the VLAN Ingress Rules (Acceptable Frame Type Check and VLAN Membership filtering), the VID and user priority are saved into the private area and frames are relayed to the kernel for the bridging process. In addition to the VID and User Priority data, the module also calculates a signature and checksum and stores this information in the private area to help ensure data integrity.

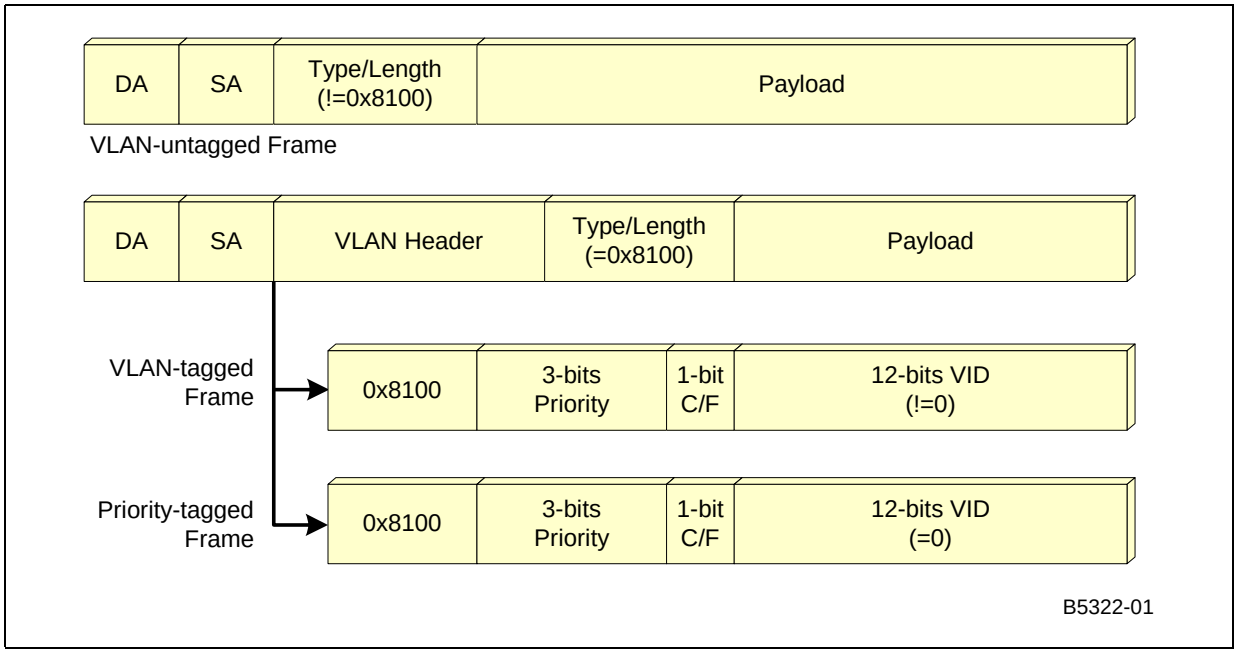
**Egress Path**

For outbound Ethernet frames, the device driver calls the API of the Egress Rules component for VLAN Egress processing if the Egress Rules component determines frames are bridged from the other NPE/Ethernet port or from an upper-layer application. If frames come from the bridge, ingress-determined VLAN Identification (VID) and user priority (both saved in the private area) are retrieved. Otherwise, the Egress Rules component calls the services of VLAN classification component to determine the VID and user priority of transmission frames. When VID and user priority of transmission frames are determined, the Database component provides the (port) member set of the VLAN group and decides whether or not the transmission port is in the member set of VLAN group. Frames are discarded if the transmission port is not in the member set of the VLAN group. If the transmission port is in the member set, egress attributes (VLAN-tagged or VLAN-untagged) of the transmission port in the VLAN group and the frame type of the outbound frames are used to determine whether or not the frame header of transmission frames should be rebuilt (insert or remove VLAN tag). After completing all egress processes, the NPE put the frames in transmission with proper headers.

## 3.1 Ingress Rules Component

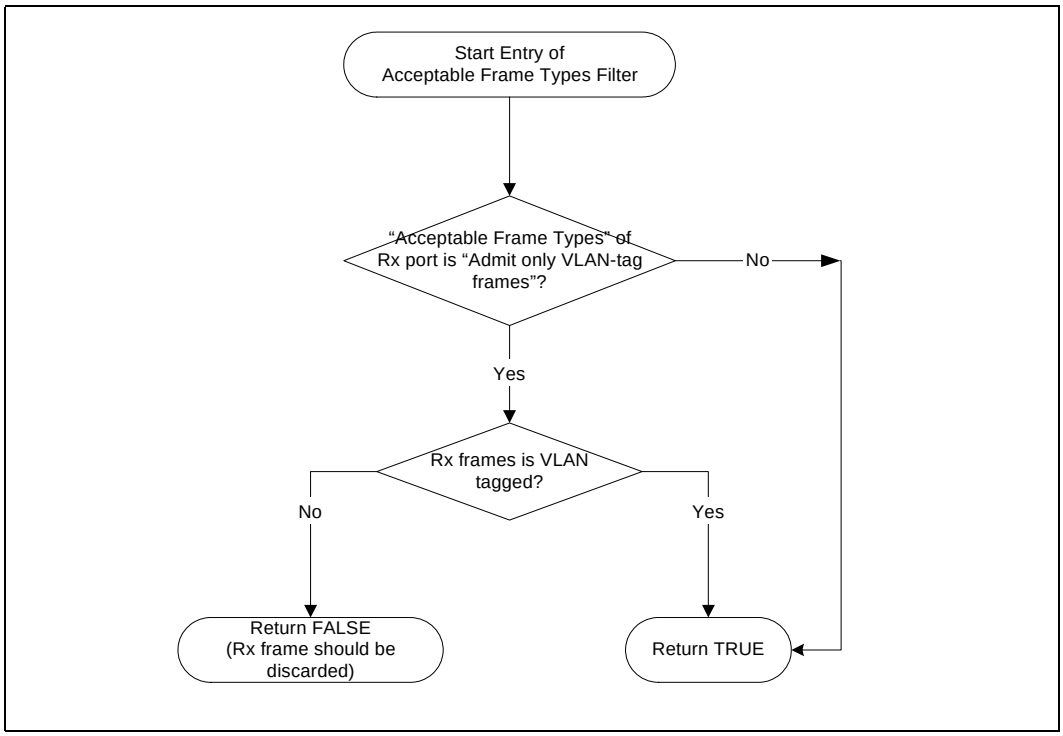
This component supports the functionality of IEEE 802.1Q Acceptable Frame Types and Ingress VLAN Membership filtering. In the IEEE 802.1Q standard, three frame types are defined — VLAN-tagged, priority-tagged, and non-VLAN-tagged. Figure 4 shows the frame types.

**Figure 4. 802.1Q Frame Types**



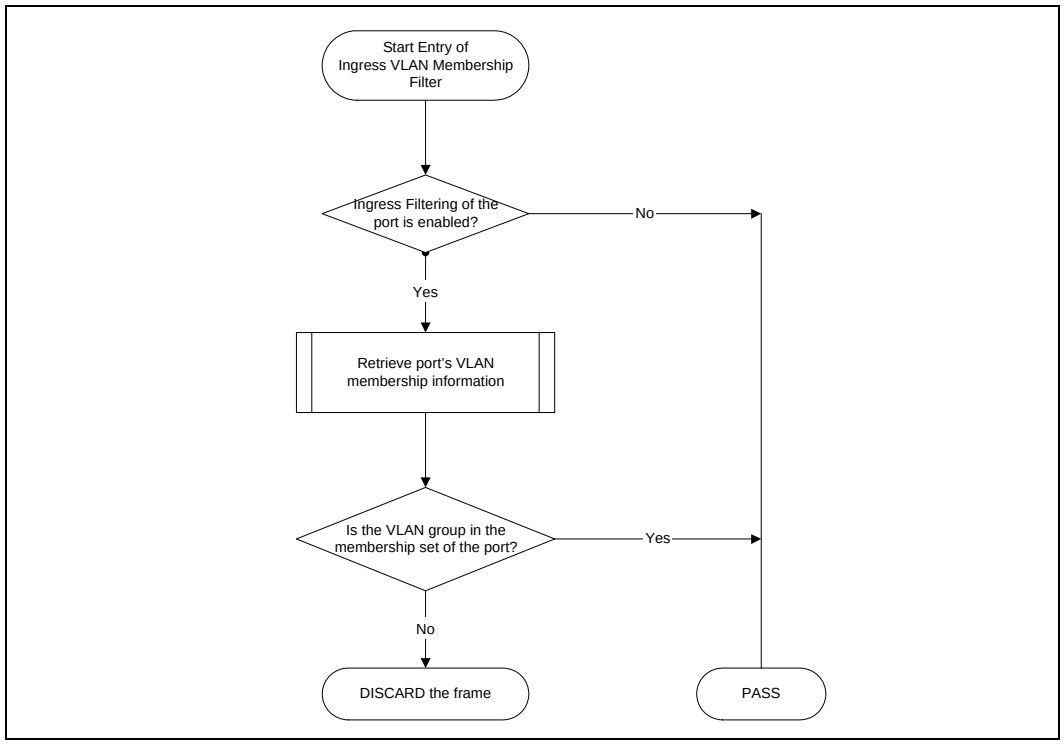

The Acceptable Frame Types parameter associated with each port controls the reception of the types of frames on that port. Valid values for this parameter are: “Admit Only VLAN-tag Frames” and “Admit All Frames”. If it is set to “Admit Only VLAN-tag Frames”, any frames received on that port which do not contain VID tagging information (i.e., untagged frames and priority-tagged frames) are discarded. Acceptable Frame Type filtering is presented in Figure 5.

**Figure 5. Flow Diagram for Acceptable Frame Type Filtering**




Ingress VLAN Membership filtering discards any frames whose VLAN group is not included in the member set of the port they were received from. This is shown in Figure 6.

**Figure 6. Flow Diagram for Ingress VLAN Membership Filtering**




### 3.1.1 External Interactions and Dependencies

The Ethernet device driver utilizes this component to perform IEEE 802.1Q Ingress Rules (Acceptable Frame Types and VLAN Membership) filtering. The Ingress rule depends on whether or not Ingress Filtering is enabled, Acceptable Frame Types parameters, and the member set of the VLAN group.

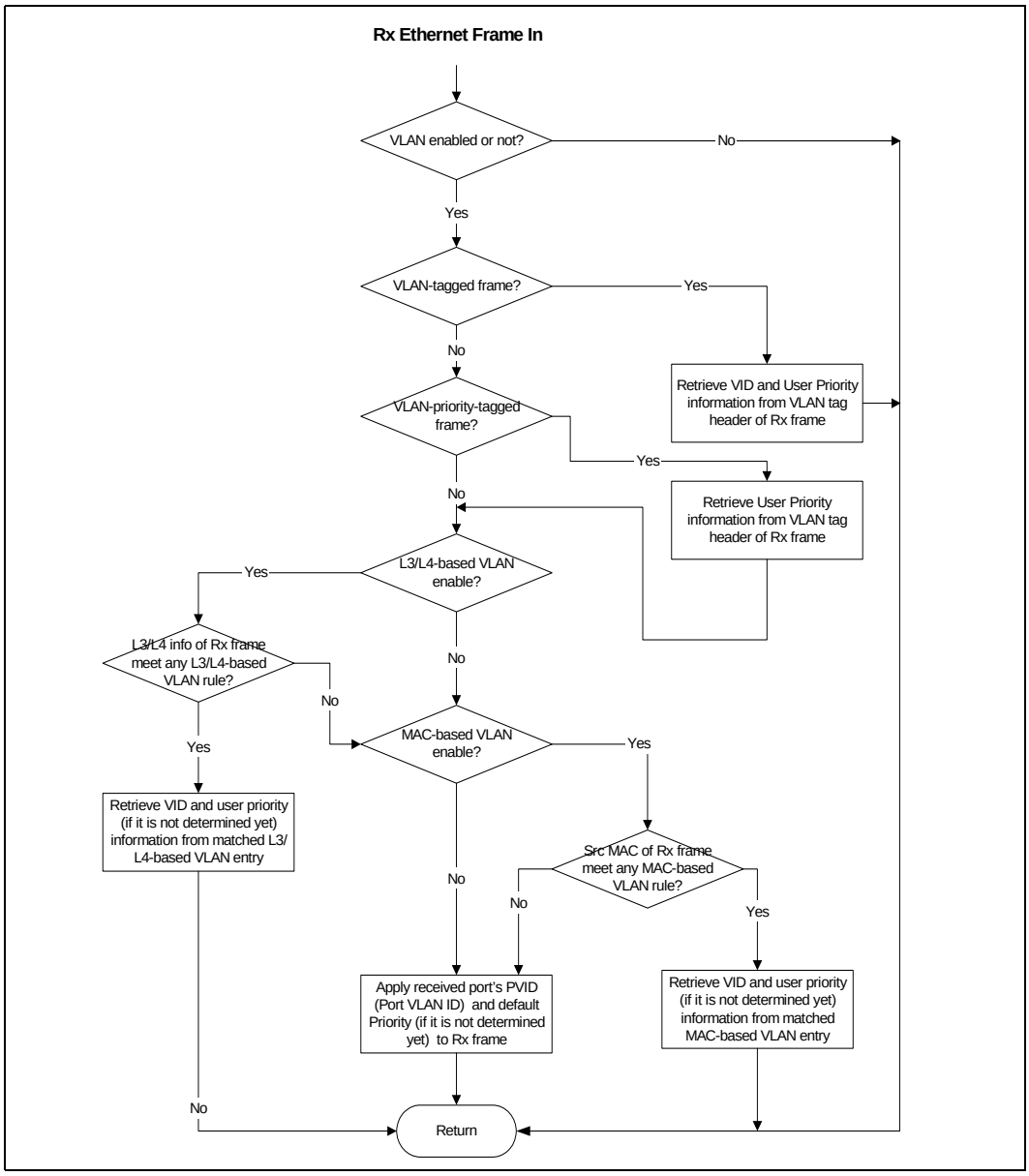
### 3.1.2 Key Assumptions

- The Database component in VLAN module is initialized and available for query.
- Default value of Acceptable Frame Types parameter for all ports is “Admit All Frames”.

## 3.2 VLAN Classification Component

VLAN Classification component is used to determine VLAN Identification (VID) and User Priority of reception/transmission frames in accordance with established classification rules. Four kinds of classification rules are supported: 802.1Q tag-based, port-based, MAC-based and Protocol (Layer 3/4) -based classifications. 802.1Q tag-based classification determines VID and priority from the VLAN-tag header of received frames. Port-based classification uses the reception port transmission port of frames to decide VID and priority. MAC-based classification uses source MAC address, and Protocol (Layer 3/4) -based classification uses information in IP/UDP/TCP IPv6/UDP/TCP/AH/ESP headers to determine a frame’s VID and priority. VLAN Classification flow is presented in Figure 7.

**Figure 7. Flow Diagram for VLAN Classification**




## 3.2.1 External Interactions and Dependencies

The Ethernet device driver utilizes the VLAN Classification component to perform Ingress and Egress VLAN Classification. The driver also utilizes services in the Database component to determine if any classification rules should be applied to frames.

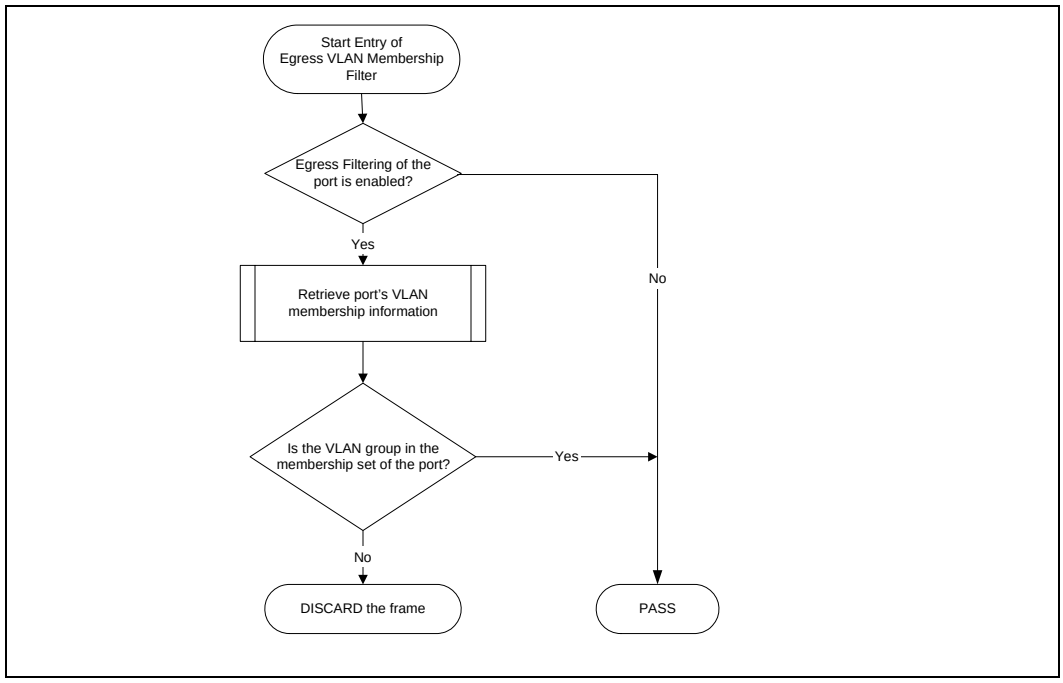
## 3.3 Egress Rules Component

This component provides functionality for IEEE 802.1Q VLAN Egress Rules. Two features are supported: Egress VLAN Membership filtering and Rebuild the Frame Header.

- Egress VLAN Membership filtering

  Discards frames whose transmission ports are not present in a frame's VID member set. This behavior is depicted in Figure 8.

**Figure 8. Flow Diagram for Egress VLAN Membership Filtering**




- Rebuilding Frame Headers

  In the IEEE 802.1Q Standard, on a given link a VLAN-aware bridge can transmit untagged frames for some VLANs and VLAN-tagged frames for others, but cannot transmit both formats for the same VLAN. A feature is provided for adding, modifying, or removing VLAN tag headers from transmission frames in accordance with tagging requirements on egress for each port. This behavior is described in Figure 9 and Table 1.

**Figure 9. Flow Diagram for Rebuilding the Frame Header**

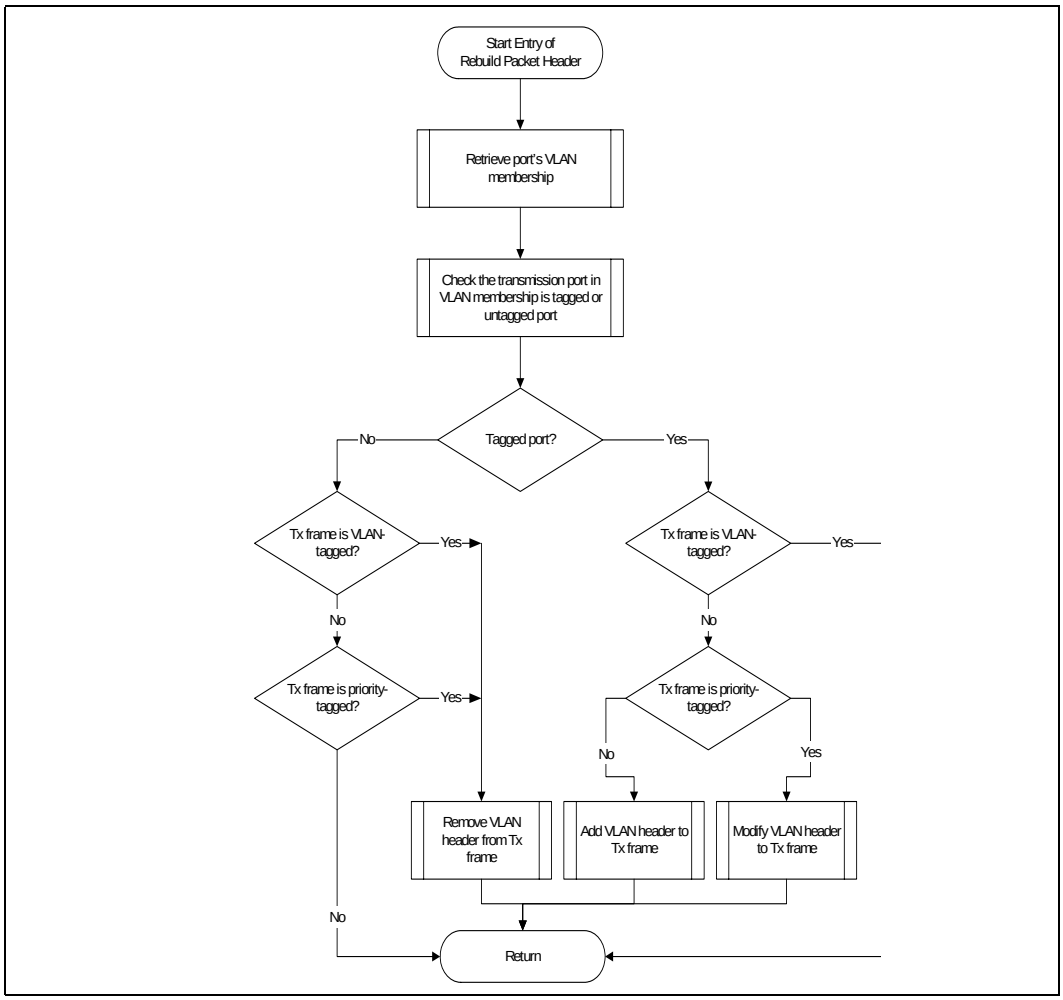



**Table 1. Rules for Rebuilding Frame Headers**

| Transmit Port Transmits Frame as: | Receive Port Receives frame as: | | |
|---|---|---|---|
| | VLAN-Tagged | Priority-Tagged | Untagged |
| untagged | Remove tag header | Remove tag header | N/A |
| VLAN tagged | N/A | Modify VLAN header | Add VLAN tag header |

## 3.3.1 External Interactions and Dependencies

The NPE Ethernet device driver utilizes this component to perform 802.1Q Egress Rules functions. It depends on whether or not Egress Filtering is enabled, and the VLAN membership of the port on transmit.

### 3.3.2 Key Assumptions

- Database component in VLAN module is initialized and available for query.

## 3.4 Database Component

This component contains all VLAN information for 802.1Q VLAN operations. There are three categories (sub-database) of information in this database: Port Database, VLAN Database, and Classification Database.

**Port Database**

This database contains information about port configurations as follows:

- PVID (Port VLAN Identification) and Default User Priority
- Status of Ingress Filtering (enable or disable)
- Acceptable frame types (AdmitAllFrames or AdmitOnlyVlanTaggedFrames)
- Port's VLAN membership table.

  ***Note:*** The VLAN membership table includes the VLAN groups to which the port belongs. Please see the IEEE 802.1Q standard for more information.

**VLAN Database**

This database contains the following information about port configurations:

- VLAN function is enabled or not.
- Egress attributes (VLAN-tagged or VLAN-untagged) of ports in VLAN group.

**Classification Rules Database**

This database contains the following classification rules:

- Rules for MAC-based classification
- Rules for Protocol (Layer 3/4) -based classification.

There are several general characteristics and functions provided by the Database component:

- Services to configure and query information of Port Database, VLAN Database, and Classification Rules Database.
- 4095 VLAN groups are supported simultaneously.
- Up to 16 MAC-based classification rules and up to 16 Protocol (Layer 3/4) -based classification rules are supported, for up to 32 simultaneous rules.

### 3.4.1 External Interactions and Dependencies

The Database component houses critical information attributes used by other sub-components of the VLAN module. Those interactions (detailed in this section) depend on which sub-database is used.

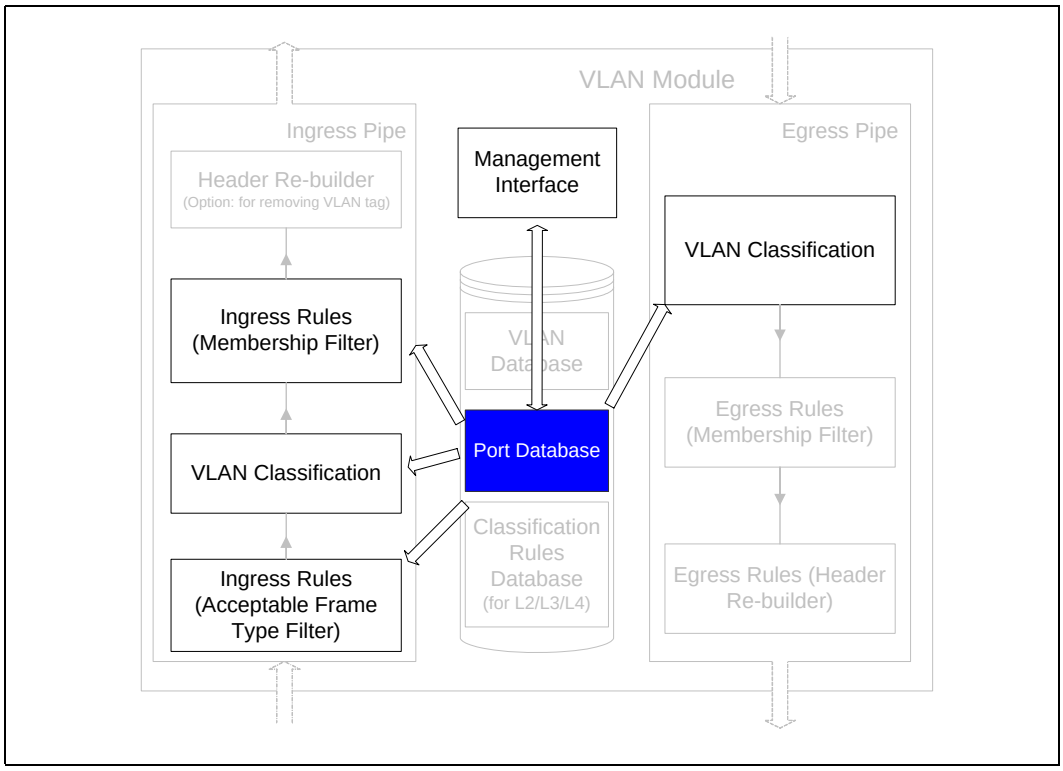
### 3.4.1.1 Port Database

Depending on the desired service, the Port database is used by the following sub-components:

- Ingress Rules: Ingress Filtering and Acceptable Frame Types attributes for each port.
- Port Based VLAN: The PVID and Default User Priority for each port.
- Port VLAN membership: Ingress Filtering and Egress Filtering.
- Management Interface: Supports interactions from the component APIs.

This is shown in Figure 10.

**Figure 10. Port Database Dependencies**

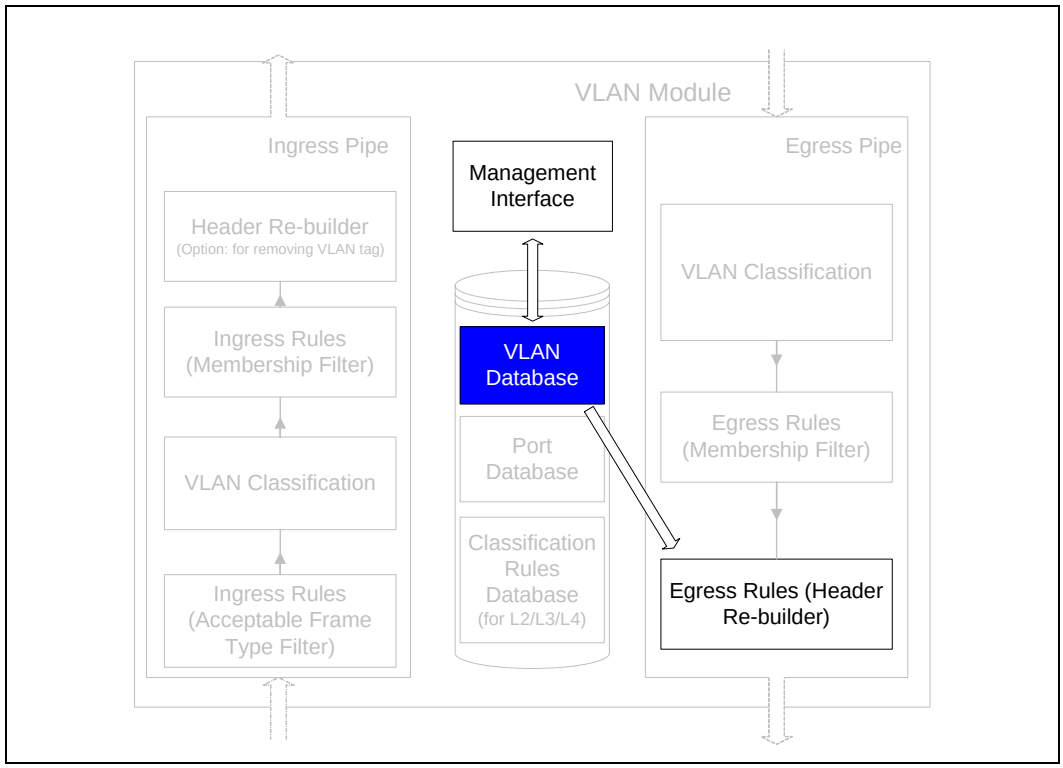



### 3.4.1.2 VLAN Database

For the services described, the VLAN database is used by the following sub-components:

- Egress Rules: Egress attributes (tagged or untagged) for egress ports in a VLAN group.
- Management Interface: Supports interactions from the component APIs

**Figure 11. VLAN Database Dependencies**

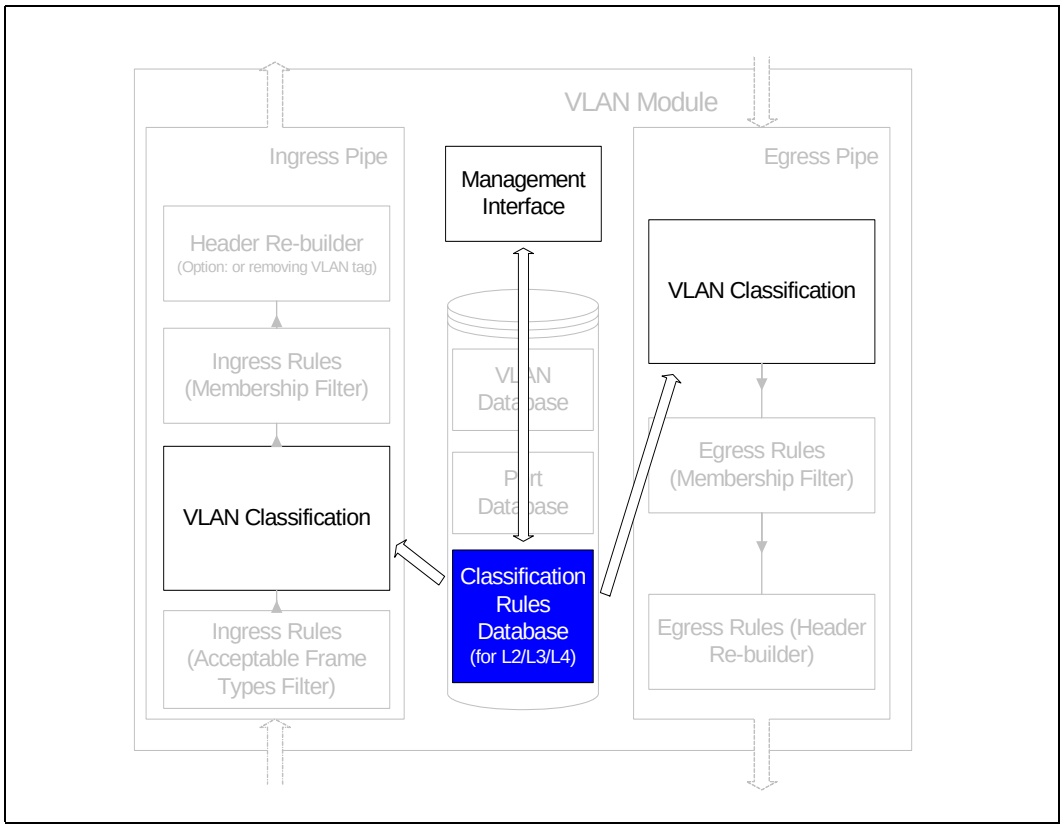



## 3.4.2 Classification Rules Database

Depending on the desired service, the Classification Rules database is used by the following sub-components:

- VLAN Classification: VID and User Priority of frames.
- Management Interface: Supports interactions from the component APIs.

**Figure 12. Classification Rules Database**




## 3.5 Management Interface Component

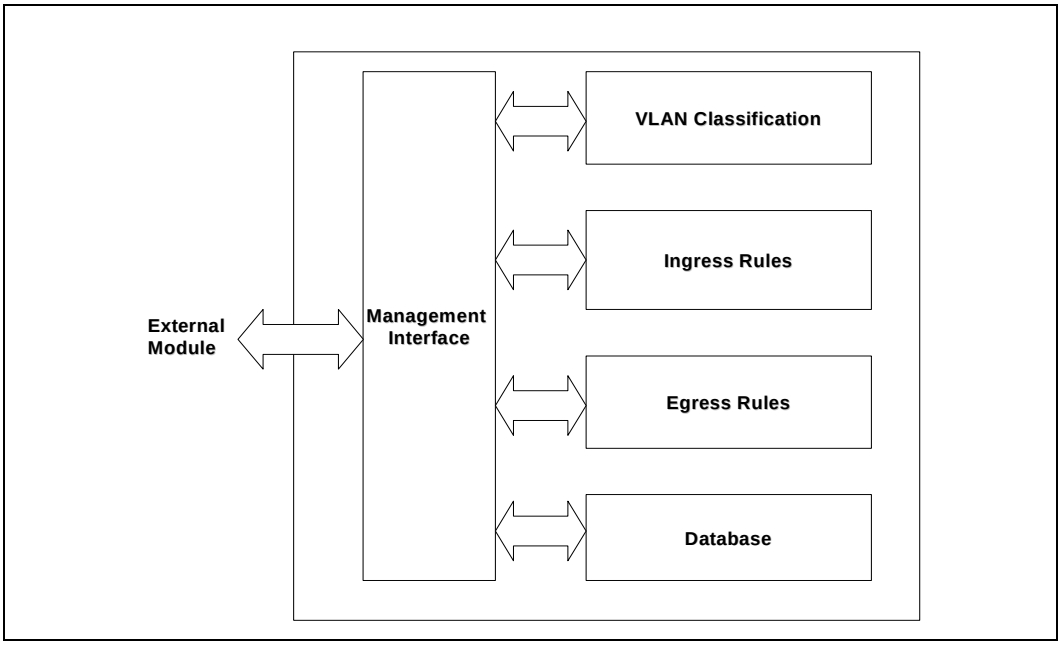
This component provides a unique interface (i.e., control path) for external modules to configure the behavior of the 802.1Q VLAN module. For example, the IOCTL parser in the Ethernet device driver should utilize this interface to access services in 802.1Q VLAN module. Direct accesses to services (or APIs) in other components in this module are not supported.

The features provided by the Management Interface component are:

- Enable & Disable 802.1Q VLAN function
- Assignment VLAN membership and associated (tagged/untagged) attributes of egress ports
- Set port's PVID and Default User Priority
- Configure Acceptable Frame Types filtering of reception port
- Enable & Disable Ingress (VLAN) Membership filtering
- Enable & Disable Egress (VLAN) Membership filtering
- Enable & Disable MAC-based VLAN Classification
- Enable & Disable Protocol (Layer 3/4) -based VLAN Classification
- Configure MAC-based VLAN Classification Rules

- Configure Protocol (Layer 3/4) -based VLAN Classification Rules.

**Figure 13. Management Interface Interactions**




# 4.0 802.1p User Priority and QoS Module

This purpose of this software module is to implement IEEE 802.1p User Priority to Traffic Class Mappings, and QoS functionality for both Ingress and Egress sides. According to IEEE 802.1p, there are a maximum of eight traffic classes supported. This module support four traffic classes: traffic class 0, 1, 2 and 3. The traffic class which is higher in numerically has the higher priority. The determination for the traffic class is performed by a combination of two subsystems: the VLAN Classifier module that provides the priority field in VLAN tag, and by the Ingress QoS - Priority Mapping Module that maps port number and VLAN priority to a traffic class. Each traffic class has its corresponding shaper with the private configuration. Depending on the shaper of frame's traffic class, the traffic could be forwarded to the next module, buffered in a priority queue, or get dropped.

For each shaper, there are two types of shapers: Data Bytes shaper (D-type), and Frame Count shaper (F-type). There are two parameters associated with each shaper: Average Rate (avgD/avgF), and Ceil Rate (ceilD/ceilF). The shaper is designed with the concept of a token bucket. The shaper design controls the long-term rate of traffic while also allowing some short-term bursts. Because the D-type shaper monitors the bandwidth of a certain traffic class, they are used extensively in the network community. The purpose for an F-type shaper is to control the number of frames allowed for further processing. The F-type shaper is commonly used to limit the number of table lookups required by the host CPU, which is often a system bottleneck. The upper-layer user interface, however, can be configured to use both shapers simultaneously, one of the two shaper types at a time, or disable the shaper for some particular traffic class. When the shaper for a traffic class is disabled, frames belonging to that traffic class are passed to the next module directly.

When the shaper type (D-type, F-type, or both) is enabled for a traffic class, frames that are classified to this traffic class have to go through its corresponding shaper. When the corresponding shaper still has available quota and no frames are buffered in the frame queue, the frame is passed to the next module. Alternately, if any of the enabled shapers is over-quota, then the frame is sent to frame queue for that traffic class. Before that frame is physically pushed to a frame queue, the queue length limitation corresponding to that traffic class is examined. If the queue length exceeds the limitation, the frame is dropped instantly, otherwise the frame is pushed to the queue.

Periodically, the timer module is triggered and updates shapers that are enabled. After updating all enabled shapers, a signal is sent to the priority queue service routine, which pops frames from the high-priority traffic class queue to the low-priority traffic queue. For each frame at the head of a certain queue, the shaper status is checked before sending to the next module. If the shaper has available queues, the frame is popped and sent to the next module, after that the corresponding shapers are updated. If the shapers are already over-quota, then the priority queue service routine goes to the next queue for service.

**802.1p Priority Mapping and Ingress QoS Sub-Components**

- Traffic Shaper

  Traffic rate control and frame queue for buffering frames.

- User Priority to Traffic Class Mapping

  Processes for IEEE 802.1Q/p User Priority to Traffic Class Mappings is handled in this component.

- Management Interface

  Control path for maintaining associated database as well as shaper configuration.

The modularization of 802.1p User Priority to Traffic Class Mappings and QoS components (such as ingress queues and traffic shaper) makes it easy for updating/enhancing Ingress QoS functions. No effort is required for design changes in 802.1p User Priority to Traffic Class Mappings when a new queueing discipline is added into the ingress queue module, or when a new traffic control algorithm is defined for the traffic shaper.

## 4.1 Traffic Shaper Component

The purpose of the Traffic Shaper component is to control the rate of traffic sent to the next software module. In general, the higher traffic class is treated as higher priority traffic. The Traffic Shaper determines if the frame should be passed to next module directly, queued in the ingress priority queues, or if the frame should be dropped. As previously stated, frames are sent to the priority queue under the condition that either traffic rate exceeds the shaper's configured rate or there are frames waiting in the corresponding priority queue. If rate limitation is not exceeded, and there are no frames buffered in the respective queue, frames are relayed to the next module. In the case where the frame must be delayed, the module first checks the number of queued frames of the particular queue. If the buffered length is above the limitation, then the frame is discarded; otherwise frames are pushed to the corresponding queue. If shapers for a traffic class are disabled, then traffic classified to this traffic class is treated as rate-unlimited. Frames are passed to the next module directly.

Each time a frame is sent to the next software module, the shapers are updated accordingly. If both D-type and F-type shapers are enabled, then the quota for both types are updated. If only D-type is enabled, then only the quota for D-type is updated. The F-type shaper quota update process is the same.

Periodically, the timer (interrupt service routine) updates the quota of token buckets by the parameters of shaper configuration. The shaper parameters are:

- Average data rate in bps (bytes per second)
- Average packet count in fps (frames per second)
- Ceil data rate in bps
- Ceil packet count in fps
- Type of shaper: D-type or F-type

The timer parameters are:

- Period of timer in millisecond (as compiler option)

The functionality provided by the Traffic Shaper component includes:

- Initialize Traffic shaper and unload traffic shaper
- Determine if the traffic conforms to the shaper configuration
- Configure the traffic rate passing through the shaper component by:
  - rate in frames
  - rate in bytes
- Queuing the frames once the traffic rate exceeds the specified rate of the shaper.

### 4.1.1 External Interactions and Dependencies

This software component is initialized and configured by the Management Interface component. as described in the following steps. The Traffic Shaper component utilizes services to retrieve frames from the Ingress or Egress thread and to push them to the next stage.

1. The device driver (via the Management Interface component) calls the API to initialize whole QoS modules, including the Traffic Shaper component.
2. The device driver calls the API to register the callback function.
3. The device driver calls the API to initiate the QoS process.
4. The software has the capability to determine the traffic class of a particular traffic. It utilizes the VLAN classification to determine the user priority of the frame and maps the user priority into a traffic class. If the number of frames of the traffic class does not exceed the configured rate of its corresponding shaper, execute the next step. Otherwise, skip to step 6.
5. Call the registered Rx callback function or CSR submitting function to relay frames to next stage in software. The process is now complete.
6. Check the queued length. If there is room in the buffered queue, proceed to step 7. Otherwise drop this frame. The process is now complete.
7. Push the frame into the buffered queue.
8. Queue service routine is executed by timer ISR and dequeue frames from queues.

## 4.2 Priority Mapping Component

This component maps the user priority (0~ 7) (determined in VLAN classification) of a received frame into the corresponding traffic class value. Frames are classified into different classes and call the services of the Traffic Shaper component to determine if frames should be queued or not. This process is presented in Figure 14.

**Figure 14. 802.1p User Priority to Traffic Class Mapping**

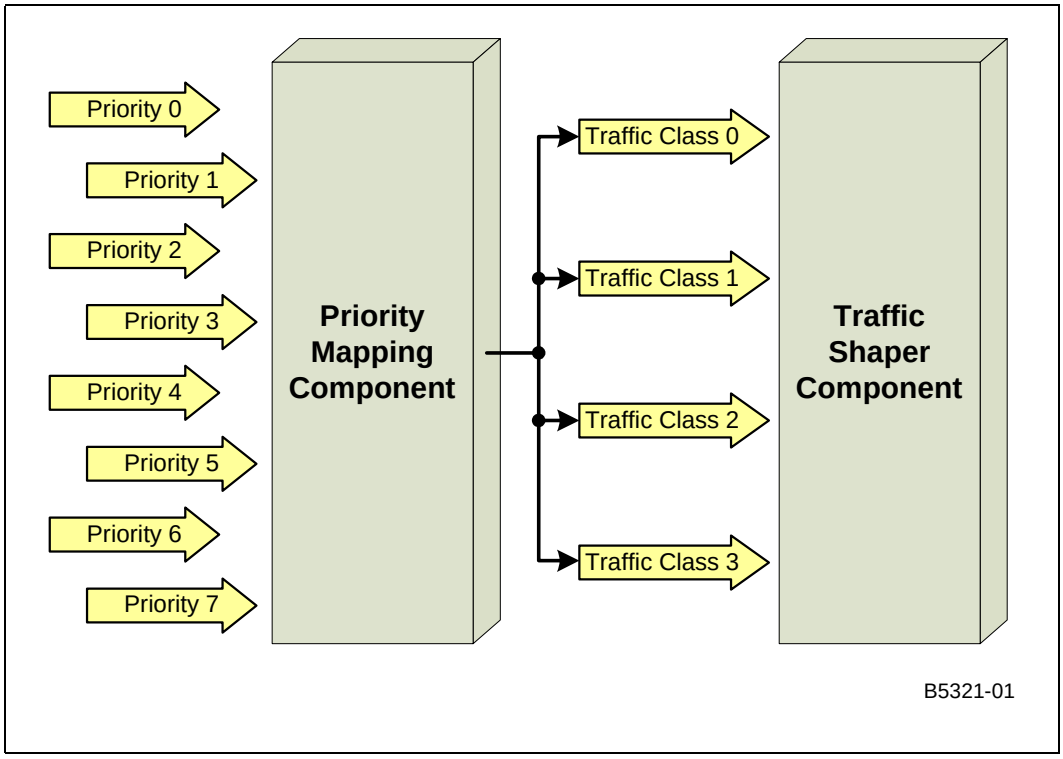



**Table 2. User Priority to Traffic Class Defaults and Recommendations**

| User Priority | Traffic Class (Default) | Traffic Class (Recommendation) |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 2 | 0 | 0 |
| 3 | 0 | 0 |
| 4 | 0 | 0 |
| 5 | 0 | 1 |
| 6 | 0 | 2 |
| 7 | 0 | 3 |

The functionality provided by the Priority Mapping component includes:

- Support 802.1Q/p User Priority to Traffic Class Mappings of received and transmitting frames.
- Interface to configure the table for mapping from User Priority to Traffic Class.

### 4.2.1 External Interactions and Dependencies

This software component is initialized and configured by the Management Interface component. The device driver utilizes services of this component for the QoS module. The Traffic Shaper component uses the Priority Mapping component to determine the traffic class of received or submitting frames.

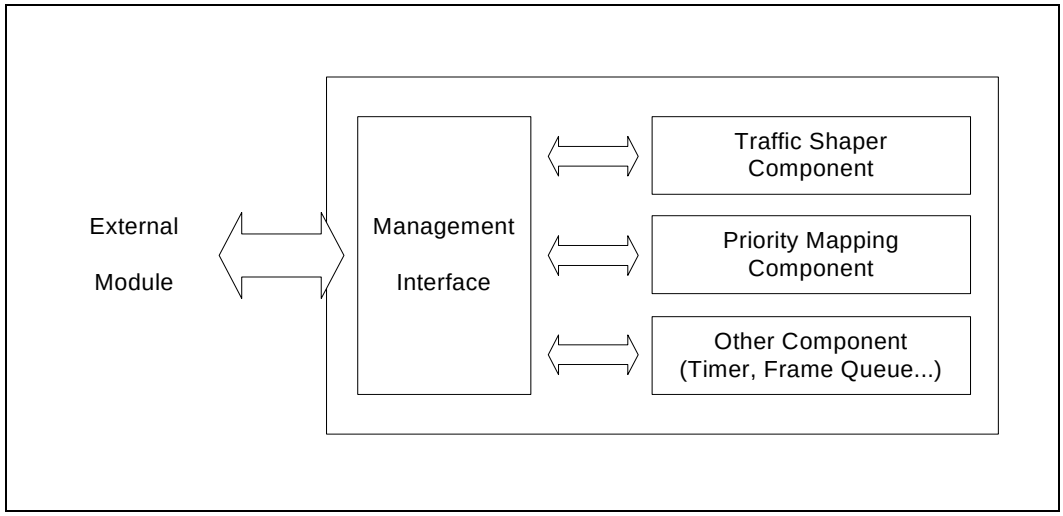
### 4.2.2 Key Assumptions

- Maximum number of User Priority is defined 802.1Q/p and the traffic class support by QoS module is designed to be four by implementation.
- Each Traffic class is associated with a traffic shaper; default state of a shaper is disabled as QoS module is initialized.

## 4.3 Management Interface Component

This component provides the public interface (i.e., control path) for external modules to configure the behavior of 802.1p User Priority to Traffic Class Mappings and QoS module. The IOCTL parser in Ethernet device driver should utilize this interface to access services of the QoS module. Direct access to services (or APIs) in other components of this module is not supported. These Interactions are shown in Figure 15.

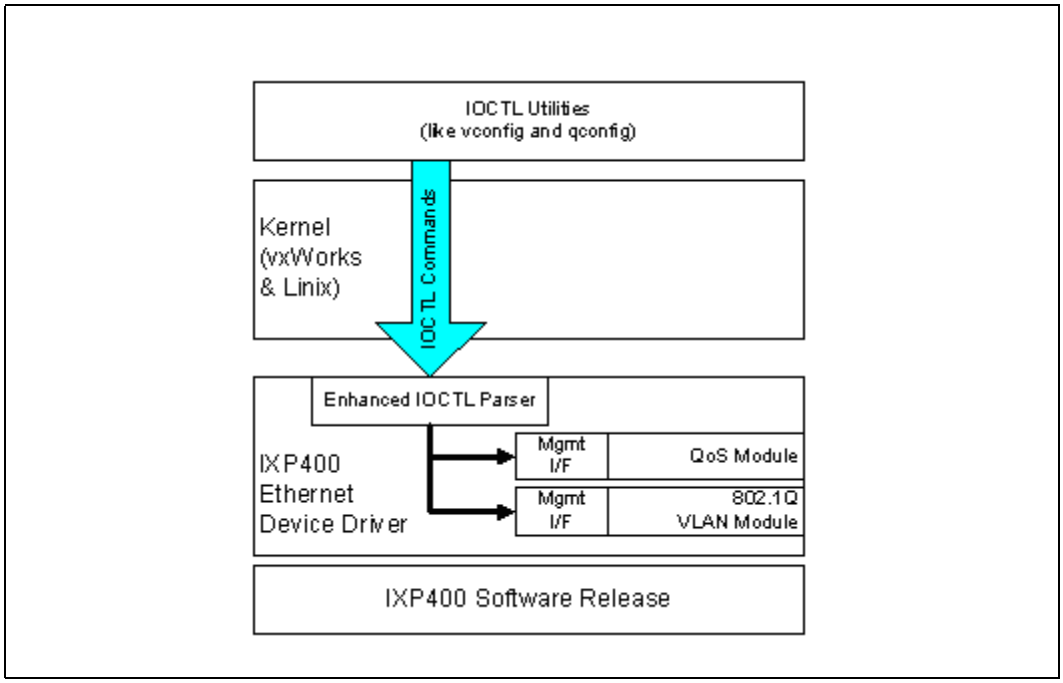
**Figure 15. Interactions of the QoS Module Management Interface Sub-Component**



# 5.0 IOCTL Enhancements for Ethernet Drivers

The purpose of this software is to extend existing IOCTL functionality in the IXP400 Ethernet device drivers. New IOCTL commands are defined to support new features for 802.1Q VLAN and QoS Modules. These commands are grouped into a configuration utility named **vqconfig**. The utility communicates with the common module to access VLAN and QoS services in the IXP400 Ethernet device driver. The IOCTL Parser recognizes these IOCTL commands and in turn executes associated services in the VLAN and QoS modules. The system view is presented in Figure 16

**Figure 16. System View of IOCTL Utilities and Parser**




The **vqconfig** utility sends management API calls via IOCTL commands to the VLAN and Qos Module. More detailed information regarding the syntax of the utility is available in the *Intel® IXP400 Software: VLAN and QoS Application Version 2.0 Release Notes*.

# 6.0 API Reference

**Table 3. API Index (Sheet 1 of 2)**

**Table 3. API Index (Sheet 2 of 2)**



## 6.1 Data Type Definitions

This section contains the data type definitions and data structure descriptions that will be used in the VLAN and QOS application programmer interface.

**Direction Type for Data Path**

```
typedef enum  {
    IX_VLAN_QOS_INGRESS = 0,
    IX_VLAN_QOS_EGRESS
}   IxVlanQosDirection;
```

Description:

Define the identifiers to be used as a function parameter for specifying the direction in the data path.

**Status Type**

```
typedef enum  {
    IX_VLAN_QOS_SUCCESS = 0,
    IX_VLAN_QOS_FAIL
}   IxVlanQosStatus;
```

Description:

Define the identifiers to be used as the status value returning from calling function.

**Acceptable Frame Type**

```
typedef enum {
    ACCEPT_TAGGED_ONLY,
    ACCEPT_ALL_FRAME,
}   IxVlanAcceptbaleFrameType;
```

Description:

Define the identifiers to be used as a function parameter for specifying the acceptable frame type.

**Egress Frame Type**

```
typedef enum {
    TAGGED_FRAME_TYPE,
    UNTAGGED_FRAME_TYPE,
}   IxVlanEgressType;
```

Description:

Define the identifiers to be used as a function parameter for specifying the frame type as egress.

**Data Types of Rule Content**

```
typedef unsigned char  MAC_ADDRESS[6];
typedef unsigned char  IP_ADDRESS[16];
typedef signed   long  PROTOCOL_TYPE;
typedef signed   long  TC;
typedef signed   long  PORT_NUM;
typedef unsigned long  SPI;
```

Description:

Define data types which are used in the content of a rule for MAC-Based VLAN classification or Protocol-Based VLAN classification.

Definition:

***MAC_ADDRESS***: MAC address of an Ethernet frame.

***IP_ADDRESS***: The IP address of a frame in IP or IPv6 protocol types. If it is used for specifying an IPv4 address, only the first 4 bytes are used and the rest bytes will not be considered.

***PROTOCOL_TYPE***: This carries the layer 3 and layer 4 protocol types.

***PORT_NUM***: Port number in a TCP or UDP header. The range for valid value is from 0 to 65535.

***TC***: Traffic class specified in an IPv6 header. The range for valid value is from 0 to 255.

***SPI***: Security Parameter Index in an AH or an ESP header. The numbers within the range which be represented by a 32 bits unsigned integer are all valid for SPI.

**Data Type for VLAN and QoS functionality**

```
typedef unsigned short RULE_ID;
typedef UINT32 IxQosTcId;  /* valid values: 0, 1, 2, 3 */
typedef UINT32 IxVlanQosPortId;
typedef UINT32 IxVlanQosPriority;
```

Definition:

***IxQosTcId***: Identifier of traffic class for QoS. The valid values are 0, 1, 2, and 3.

***IxVlanQosPortId***: Identifier of a NPE Ethernet port.

***IxVlanQosPriority***: User priority defined in IEEE802.1Q

***RULE_ID***: Identifier of a rule for MAC-Based or Protocol-Based VLAN classifications.

**Data Structure for MAC Rule**

```
typedef struct  {
    UINT32  vid;
    UINT32  priority;
    MAC_ADDRESS src_mac;
}   IxVlanMacRule;
```

Description:

Define the data structure be used as the function parameter to specify a rule for MAC-Based

VLAN classification. The frame whose source MAC address is identical to the MAC address specified in the rule is called it matches to the rule. The VLAN group of the frame which matches to the rule is assigned to the VLAN ID specified in the rule.

Structure member:

***vid***: Specifying the VLAN ID for the frames which match to the rule.

***priority***: Specifying the 802.1Q user priority for the frames which match to the rule.

***src_mac***: The MAC address used to match to source MAC address of an Ethernet frame.

**Data Structure for Protocol Rule**

```
typedef struct  {
    UINT32  vid;
    UINT32  priority;
    IP_ADDRESS src_ip, src_ip_mask;
    IP_ADDRESS dst_ip, dst_ip_mask;
    PROTOCOL_TYPE protocol;
    TC          tc;
    PORT_NUM   src_port, src_port_end;
    PORT_NUM   dst_port, dst_port_end;
    SPI         spi;
}   IxVlanIpRule;
```

Description:

Define the data structure be used as the function parameter to specify a rule for Protocol-Based VLAN classification. The rule describes an Ethernet frame by specifying the ranges of value for some fields of protocol headers of a frame in layer 3 and layer 4. The frame which is identical to the describing by the rule is called it matches to the rule. The VLAN group of the frame which matches to the rule is assigned to the VLAN ID specified in the rule.

Structure member:

***vid***: Specifying the VLAN ID for the frames which match to the rule.

***priority***: Specifying the 802.1Q user priority for the frames which match to the rule.

***src_ip, src_ip_mask***: The IP address and mask used to specifying a range of the source IP address for an IP or IPv6 protocol frame.

***dst_ip, dst_ip_mask***: The IP address and mask used to specifying a range of the destination IP address for an IP or IPv6 protocol frame.

***protocol***: This specifies the protocol type of an Ethernet frame. The high word specifies the protocol type in layer 3. It shall use the identifiers ETH_IP and ETH_IPV6 to represent IP and IPv6 protocol types respectively. The low word specifies the protocol type in layer 4. It shall be filled with the value of protocol type directly. The software only supports TCP and UDP for an IP protocol frame and support TCP, UDP, ESP and AH for an IPv6 frame.

***tc***: This specifies the traffic class in the IPv6 protocol header. Use UNSPECIFIED_TC to indicate software not to care this field.

***src_port***: This specifies the number of source port for a TCP or UDP header. Use UNSPECIFIED_PORT for src_port indicates software not to care this field.

***src_port_end***: This specifies the end of a range of port numbers which starting with src_port. Use UNSPECIFIED_PORT to indicate software none or a single port is specified by

src_port.

***dst_port***: This specifies the number of destination port for the TCP or UDP header. Use UNSPECIFIED_PORT for dst_port indicates software not to care this field.

***dst_port_end***: This specifies the end of a range of port numbers which starting with dst_port. Use UNSPECIFIED_PORT to indicate software none or a single port is specified by dst_port.

***spi***: This specifies the value of Security Parameter Index for the ESP or AH header. Use UNSPECIFIED_SPI to indicate software not to care this field.

**Data Type for Call Back Functions**

```
typedef void (*IxVlanQosCallbackFn)(UINT32 callbackTag,
                                    IX_OSAL_MBUF *buffer);

typedef void (*IxVlanResvBufGetCallbackFn)(UINT32 callbackTag,
                                           IX_OSAL_MBUF *buffer,
                                           UINT8 **retBuffer);
```

Definition:

***IxVlanQosCallbackFn***: The callback function used to drop an Ethernet frame. Once the VLAN and QoS modules discard a TX or RX frame, the module calls this function, which is registered by Ethernet driver or client software.

***IxVlanResvBufGetCallbackFn***: The callback function used to retrieve a reserved buffer once the VLAN modules save the VLAN information of a received frame. This callback function is offered and registered by the Ethernet driver or client software. The allocation for reserved spaces and maintenance of the linkage to each MBUF is the responsibility of the software that registered this callback function.

## 6.2 Function Prototype Definitions

This section contains the VLAN and QoS Example Code APIs and data structures.

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixVlanQosModuleInitialize** (void); | |
| **Parameters:** | **Description:** | **I/O:** |
| n/a | None | |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on initialization.<br>IX_VLAN_QOS_FAIL: Fail on initialization. | |
| **Description:** | Initialize the VLAN and QoS modules. This function is called when the client software or Ethernet driver, which use the VLAN and QoS modules, is started. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixVlanQosModuleUninitialize** (void); | |
| **Parameters:** | **Description:** | **I/O:** |
| n/a | | |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on un-initialization.<br>IX_VLAN_QOS_FAIL: Fail on un-initialization. | |
| **Description:** | Un-initialize the VLAN and QoS modules. This function is called when the client software or Ethernet driver, which use the VLAN and QoS modules, is exiting or being removed. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixVlanQosPortTxFrameSubmit** (IxVlanQosPortId ***portId***, IX_OSAL_MBUF ****buffer***, UINT32 ***priority***); | |
| **Parameters:** | **Description:** | **I/O:** |
| portId | The identifier of the NPE Ethernet port to transmit Ethernet frame on. | I |
| buffer | Address of an MBUF which representing the Ethernet frame to be transmitted. | I |
| priority | Relative priority used to transmit a frame. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on transmission.<br>IX_VLAN_QOS_FAIL: Failed on transmission. | |
| **Description:** | This function shall be used to submit MBUFs buffers for transmission on a particular MAC device. Software shall use this function instead of ixEthAccPortTxFrameSubmit to applying processing of VLAN and QoS functionality for the Egress frames. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixVlanQosPortRxCallbackRegister** (IxEthAccPortId ***portId***, IxEthAccPortRxCallback ***rxCallbackFn***, UINT32 ***callbackTag***); | |
| **Parameters:** | **Description:** | **I/O:** |

| | | |
|---|---|---|
| portId | The identifier of the NPE Ethernet port the callback is registered to. | I |
| rxCallbackFn | Function to be called when Ingress Ethernet frames are received. | I |
| callbackTag | This tag shall be provided to the callback function. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on registration.<br>IX_VLAN_QOS_FAIL: Failed on registration. | |
| **Description:** | Register a callback function to allow the reception of frames. The registered callback function is called once a frame is received by this service. This function is used to replace ixEthAccPortRxCallbackRegister when software expects the processing for VLAN and QoS functionality to be applied on Ingress frames. | |

| Prototype: | IxVlanQosStatus **ixVlanQosPortTxDropCallbackRegister** (IxEthAccPortId ***portId***, IxVlanQosCallbackFn ***txDropCallbackFn***,UINT32 ***callbackTag*** ); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| portId | The identifier of the NPE Ethernet port the callback is registered to. | I |
| txDropCall-backFn | Function to be called when Egress frames are discarded. | I |
| callbackTag | This tag shall be provided to the callback function. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on registration.<br>IX_VLAN_QOS_FAIL: Failed on registration. | |
| **Description:** | Register a callback function to drop an Egress frame requested either by VLAN or QoS modules. The registered callback function is called once a frame being transmitted is to be dropped by this software. | |

| Prototype: | IxVlanQosStatus **ixVlanQosReservedBufferGetCallbackRegister** (IxVlanQosPortId ***portId***, IxVlanResvBufGetCallbackFn ***callbackFn***, UINT32 ***callbackTag*** ); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| portId | The identifier of the NPE Ethernet port the callback is registered to. | I |
| callbackFn | Function to be called when the reserved buffer is requested. | I |
| callbackTag | This tag shall be provided to the callback function. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on registration.<br>IX_VLAN_QOS_FAIL: Failed on registration. | |
| **Description:** | Register a callback function to retrieve a reserved buffer associated with a MBUF. The software uses this buffer to save the VLAN information of a frame which represented by a MBUF. The VLAN information is determined during the VLAN processing is applying on the received frame. Once the frame has been bridged to another NPE Ethernet port, the VLAN information will be extracted from the buffer by software on Egress.<br>**NOTE:** The client software or Ethernet driver shall reserve 16 bytes space for each MBUF. | |

| Prototype: | IxVlanQosStatus **ixVlanPortEnable** (IxVlanQosPortId ***pid***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | The identifier of the NPE Ethernet port to be enabled. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on enabling the port.<br>IX_VLAN_QOS_FAIL: Fail on enable the port. | |
| **Description:** | Enable the VLAN functionality for a given NPE Ethernet port. The functionality is disabled by default. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixVlanPortDisable** (IxVlanQosPortId ***pid***); | |
| **Parameters:** | **Description:** | **I/O:** |
| pid | The identifier of the NPE Ethernet port to be enabled. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on disabling the port.<br>IX_VLAN_QOS_FAIL: Fail on disabling the port. | |
| **Description:** | Disable the VLAN functionality for a given NPE Ethernet port. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixVlanPortEnabledGet** (IxVlanQosPortId ***pid***, BOOL ****enabled***); | |
| **Parameters:** | **Description:** | **I/O:** |
| pid | The identifier of the NPE Ethernet port to retrieve from. | I |
| *enabled | Address of the space used to retrieve the port is whether enabled. Zero value indicates the port is disabled and nonzero value indicates the port is enabled. The address cannot be NULL. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving status from the port.<br>IX_VLAN_QOS_FAIL: Fail on retrieving status from the port. | |
| **Description:** | Retrieve the Boolean value from a given NPE Ethernet port indicating whether the VLAN functionality on the port is enabled. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixVlanEgressTypeSet** (IxVlanQosPortId ***pid***, IxVlanVlanId ***vid***, IxVlanEgressType ***type***); | |
| **Parameters:** | **Description:** | **I/O:** |
| pid | The identifier of the NPE Ethernet port to be set. | I |
| vid | The identifier of the VLAN to be set. | I |
| type | The frame type of a frame being transmitted in. Use TAGGED_FRAME_TYPE for tagging the frames and use UNTAGGED_FRAME_TYPE for un-tagging the frames. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on setting the egress type.<br>IX_VLAN_QOS_FAIL: Fail on setting the egress type. | |
| **Description:** | Set the VLAN egress tagging or un-tagging for a given NPE Ethernet port and VLAN ID. If egress VLAN tagging is set, the untagged frame will be transmitted in a tagged format. If egress VLAN un-tagging is set, the tagged frame will be transmitted in untagged format. The egress type is un-tagging by default.<br>**NOTE:** If the VLAN ID of the frame is not joined in the membership table of the egress port, the frame will be transmitted without being changed. | |

| Prototype: | IxVlanQosStatus **ixVlanEgressTypeGet** (IxVlanQosPortId ***pid***, IxVlanVlanId ***vid***, IxVlanEgressType ****type***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | The identifier of the NPE Ethernet port to retrieve from. | I |
| vid | The identifier of the VLAN to be retrieved. | I |
| *type | Address of the space used to retrieve the egress frame type. TAGGED_FRAME_TYPE indicates the egress frames will be tagging and UNTAGGED_FRAME_TYPE indicates the egress frames will be un-tagging. The address cannot be NULL. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the egress type.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the egress type. | |
| **Description:** | Retrieve the egress type from a given NPE Ethernet port and VLAN ID. Egress type indicates the format of the egress frames to be transmitted in. | |

| Prototype: | IxVlanQosStatus **ixVlanMembershipSet** (IxVlanQosPortId ***pid***, IxVlanVlanId ***vid***, BOOL ***isMember***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | The identifier of the NPE Ethernet port to be set. | I |
| vid | The identifier of the VLAN to be set. | I |
| isMember | The Boolean value to indicate the VLAN ID whether to be the member of the NPE Ethernet port or not. Use TRUE value demands the VLAN ID joining into the VLAN membership of the port and FALSE value demands the VLAN ID leaving from the VLAN membership of the port. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on setting the membership.<br>IX_VLAN_QOS_FAIL: Fail on setting the membership. | |
| **Description:** | Demand a VLAN id to join to or leave from the VLAN membership of a given NPE Ethernet port.<br>**NOTE:** The PVID (default VLAN ID of the port) cannot leave the VLAN membership of the port. | |

| Prototype: | IxVlanQosStatus **ixVlanMembershipGet** (IxVlanQosPortId ***pid***, IxVlanVlanId ***vid***, BOOL ****isMember***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | The identifier of the NPE Ethernet port to retrieve from. | I |
| vid | The identifier of the VLAN to be retrieved. | I |
| *isMember | Address of the space to retrieve the Boolean value which indicating the membership of the VLAN ID on the port. TRUE value indicates the VID is joined in the VLAN membership of the port and FALSE value indicates not. The address cannot be NULL. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the membership.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the membership. | |
| **Description:** | Retrieve the VLAN membership of a VLAN ID from a given NPE Ethernet port. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixVlanPortAcceptFrameTypeSet** (IxVlanQosPortId ***pid***, IxVlanAcceptbaleFrameType ***type***); | |
| **Parameters:** | **Description:** | **I/O:** |
| pid | The identifier of the NPE Ethernet port to be set. | I |
| type | The acceptable frame type. Use ACCEPT_TAGGED_ONLY to accept VLAN tagged frames only and use ACCEPT_ALL_FRAME to accept all type of frames. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on setting the acceptable frame type<br>IX_VLAN_QOS_FAIL: Fail on setting the acceptable frame type. | |
| **Description:** | Set the acceptable frame type for a given NPE Ethernet port. Use ACCEPT_TAGGED_ONLY to accept VLAN tagged frames only and use ACCEPT_ALL_FRAME to accept all type of frames. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixVlanPortAcceptFrameTypeGet** (IxVlanQosPortId ***pid***, IxVlanAcceptbaleFrameType ****type***); | |
| **Parameters:** | **Description:** | **I/O:** |
| pid | The identifier of the NPE Ethernet port to retrieve from. | I |
| *type | Address of the space used to retrieve the acceptable frame type. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the acceptable frame type.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the acceptable frame type. | |
| **Description:** | Retrieve the acceptable frame type from a given NPE Ethernet port. The type can be accepting all type of frames or accepting VLAN tagged frames only. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixVlanPortMembershipFilterSet** (IxVlanQosPortId ***pid***, IxVlanQosDirection ***dir***, BOOL ***enabled***); | |
| **Parameters:** | **Description:** | **I/O:** |
| pid | The identifier of the NPE Ethernet port to be set. | I |
| dir | Which direction of data path to be set. IX_VLAN_QOS_INGRESS indicates the ingress side and IX_VLAN_QOS_EGRESS indicates the egress side. | I |
| enabled | The Boolean value indicating the filter to be enabled or disabled. TRUE value indicates enabling the filter and FALSE value indicates disabling the filter. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on setting the filter.<br>IX_VLAN_QOS_FAIL: Fail on setting the filter. | |
| **Description:** | Enable or disable the VLAN membership filter for a given NPE Ethernet port at the given direction. The Ingress and Egress VLAN membership filter of the port are both enabled by default. | |

| Prototype: | IxVlanQosStatus **ixVlanPortMembershipFilterGet** (IxVlanQosPortId ***pid***, IxVlanQosDirection ***dir***, BOOL ****enabled***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | The identifier of the NPE Ethernet port to retrieve from. | I |
| dir | Which direction of data path to be set. IX_VLAN_QOS_INGRESS indicates the ingress side and IX_VLAN_QOS_EGRESS indicates the egress side. | I |
| *enabled | Address of the space used to retrieve the Boolean value, which indicates if the filter is enabled. Zero value indicates the port is disabled and nonzero value indicates the filter is enabled. The address cannot be NULL. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the filter status.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the filter status. | |
| **Description:** | Retrieve whether the VLAN membership filter is enabled for a given NPE Ethernet port at the given direction. | |

| Prototype: | IxVlanQosStatus **ixVlanPortVlanTagSet** (IxVlanQosPortId ***pid***, IxVlanVlanId ***vid***, IxEthDBPriority ***priority*** ); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | The identifier of the NPE Ethernet port to be set. | I |
| vid | VLAN id of the default VLAN tag. | I |
| priority | User priority of the default VLAN tag. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on setting the default VLAN tag.<br>IX_VLAN_QOS_FAIL: Fail on setting the default VLAN tag. | |
| **Description:** | Set the default VLAN tag for a given NPE Ethernet port. The VLAN tag consists of IEEE 802.1Q user priority and VLAN ID. The default VLAN tag of the port is (priority=0, vid=1). | |

| Prototype: | IxVlanQosStatus **ixVlanPortVlanTagGet** (IxVlanQosPortId ***pid***, IxVlanVlanId ****vid***, IxEthDBPriority ****priority***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | The identifier of the NPE Ethernet port to retrieve from. | I |
| *vid | Address of the space used to retrieve the VLAD id of the default VLAN tag. | O |
| *priority | Address of the space used to retrieve the user priority of the default VLAN tag. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the default VLAN tag.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the default VLAN tag. | |
| **Description:** | Retrieve the default VLAN tag from a given NPE Ethernet port. The VLAN tag consists of IEEE 802.1Q user priority and VLAN ID. | |

| **Prototype:** | IxVlanQosStatus **ixVlanMacRuleAdd** (IxVlanMacRule ***mac_rule***, RULE_ID ***rid***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| mac_rule | The MAC rule to be added. The content of the rule shall be filled before to be added. | I |
| rid | Address of the space used to retrieve the rule identifier if the requested adding is successful. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on adding the rule.<br>IX_VLAN_QOS_FAIL: Fail on adding the rule. | |
| **Description:** | Add a rule to the database for MAC-based VLAN classification. On success, a unique identifier that associates with the newly added rule will be returned. | |

| **Prototype:** | IxVlanQosStatus **ixVlanMacRuleDelete** (RULE_ID ***rid***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| rid | The rule identifier specifying the rule to be removed. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on removing the rule.<br>IX_VLAN_QOS_FAIL: Fail on removing the rule. | |
| **Description:** | Remove a rule from the database for MAC-Based VLAN classification. The rule identifier shall be given to specify the rule. | |

| **Prototype:** | **IxVlanQosStatus ixVlanMacRuleGet** (RULE_ID ***rid***, IxVlanMacRule ***mac_rule***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| rid | The rule identifier specifying the rule whose content to be retrieved. | I |
| *mac_rule | Address of the space used to retrieve the content of the rule. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the rule.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the rule. | |
| **Description:** | Retrieve the content of a MAC-Based VLAN classification rule specified by a given identifier. | |

| **Prototype:** | IxVlanQosStatus **ixVlanMacRuleFind** (IxVlanMacRule ***mac_rule***, RULE_ID ***rid***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| *mac_rule | Address of the rule content used to be matched to the rules in the classifier database. | I |
| *rid | Address of the space used for retrieving the identifier of the rule which matched to the given content, if existed. On fail, there will be no change on that address. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on finding the rule.<br>IX_VLAN_QOS_FAIL: Fail on finding the rule. | |
| **Description:** | Find the MAC-based VLAN classification rule from the database by matching a given content. The content of the rule shall be filled before calling the function. The member fields, vid and priority, will be ignored during the content matching. | |

| Prototype: | IxVlanQosStatus **ixVlanFirstMacRuleIdGet** (RULE_ID ***rid***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| *rid | Address of the space used to retrieve the identifier of the first rule if there is any in the database. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the rule identifier.<br>IX_VLAN_QOS_FAIL: Fail on finding the rule. | |
| **Description:** | Retrieve the first MAC-Based VLAN classification rule in the classifier database. When the user wants to retrieve all rules in the classifier, the user has to call this function at first and then call the ixVlanNextMacRuleIdGet for retrieving the consequent rules. This function will return the rule identifier of the first rule, instead of returning the content. | |

| Prototype: | IxVlanQosStatus **ixVlanNextMacRuleIdGet** (RULE_ID ***rid***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| *rid | Address of the space used to retrieve the identifier of the rule in the database which is in sequence to the previous one. The address cannot be NULL. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the rule identifier.<br>IX_VLAN_QOS_FAIL: Fail on finding the rule. | |
| **Description:** | Retrieve the MAC-Based VLAN classification rule which is consequent to the rule be retrieved by calling this function or calling ixVlanFirstMacRuleIdGet at previous. This function will return the rule identifier instead of returning the rule content. | |

| Prototype: | IxVlanQosStatus **ixVlanMacRuleHitGet** (RULE_ID ***rid***, UINT32 ****hit***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| rid | The rule identifier specifying the rule whose hit count to be retrieved. | I |
| *hit | Address of the space used to retrieve the hit counter. The address cannot be NULL. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the hit counter.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the hit counter. | |
| **Description:** | Retrieve the hit counter of a MAC-based VLAN classification rule. The hit count records the number of times the rule was able to be matched to the ingress or egress frames being classified in the data path. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixVlanMacRuleHitReset** (RULE_ID ***rid***); | |
| **Parameters:** | **Description:** | **I/O:** |
| rid | The rule identifier specifying the rule whose hit count is to be reset. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on resetting the hit counter.<br>IX_VLAN_QOS_FAIL: Fail on resetting the hit counter. | |
| **Description:** | Reset the hit counter of a MAC-based VLAN classification rule to zero. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixVlanMacRuleResetAll** (void); | |
| **Parameters:** | **Description:** | **I/O:** |
| n/a | None | |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on resetting the hit counter.<br>IX_VLAN_QOS_FAIL: Fail on resetting the hit counter. | |
| **Description:** | Reset the database of the MAC-based classification. This action will remove all rules from the database. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixVlanMacClassifierSet** (IxVlanQosPortId ***pid***, BOOL ***enabled***); | |
| **Parameters:** | **Description:** | **I/O:** |
| pid | Identifier of the NPE Ethernet port to be set. | I |
| enabled | The Boolean value indicating whether the classifier is to be enabled or disabled. TRUE value indicates enabling the classifier and FALSE value indicates disabling it. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on setting the classifier.<br>IX_VLAN_QOS_FAIL: Fail on setting the classifier. | |
| **Description:** | Enable or disable the MAC-based VLAN classification for the incoming or outgoing frames on a given NPE Ethernet port. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixVlanMacClassifierGet** (IxVlanQosPortId ***pid***, BOOL ****enabled***); | |
| **Parameters:** | **Description:** | **I/O:** |
| pid | Identifier of the NPE Ethernet port to be set. | I |
| *enabled | Address of the space used to retrieve the port is whether enabled. Zero value indicates the port is disabled and nonzero value indicates the port is enabled. The address cannot be NULL. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the classifier.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the classifier. | |
| **Description:** | Retrieve the Boolean value from a given NPE Ethernet port indicating that the MAC-based VLAN classification on the port is enabled. | |

| Prototype: | IxVlanQosStatus **ixVlanProtocolRuleAdd** (IxVlanIpRule ***ip_rule***, RULE_ID ***rid***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| *ip_rule | The Protocol rule to be added. The content of the rule shall be filled before to be added. | I |
| *rid | Address of the space used to retrieve the rule identifier if the requested adding is successful. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on adding the rule.<br>IX_VLAN_QOS_FAIL: Fail on adding the rule. | |
| **Description:** | Add a rule to the database for Protocol-Based VLAN classification. On success, a unique identifier which associates with the newly added rule is returned. | |

| Prototype: | IxVlanQosStatus **ixVlanProtocolRuleDelete** (RULE_ID ***rid***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| rid | The rule identifier specifying the rule to be removed. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on removing the rule.<br>IX_VLAN_QOS_FAIL: Fail on removing the rule. | |
| **Description:** | Remove a rule from the database for Protocol-Based VLAN classification. The rule identifier shall be given to specify the rule. | |

| Prototype: | IxVlanQosStatus **ixVlanProtocolRuleGet** (RULE_ID ***rid***, IxVlanIpRule ***ip_rule***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| rid | The rule identifier specifying the rule whose content to be retrieved. | I |
| *ip_rule | Address of the space used to retrieve the content of the rule. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the rule.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the rule. | |
| **Description:** | Retrieve the content of a Protocol-Based VLAN classification rule specified by a given identifier. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixVlanProtocolRuleFind** (IxVlanIpRule ***ip_rule***, RULE_ID ***rid***); | |
| **Parameters:** | **Description:** | **I/O:** |
| *ip_rule | Address of the rule content used to be matched to the rules in the classifier database. | I |
| *rid | Address of the space used for retrieving the identifier of the rule which matched to the given content, if existed. On fail, there will be no change on that address. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on finding the rule.<br>IX_VLAN_QOS_FAIL: Fail on finding the rule. | |
| **Description:** | Find the Protocol-Based VLAN classification rule from the database by matching a given content. The content of the rule shall be filled before calling the function. The member fields, vid and priority, will be ignored during the content matching. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixVlanFirstProtocolRuleIdGet** (RULE_ID ***rid***); | |
| **Parameters:** | **Description:** | **I/O:** |
| *rid | Address of the space used to retrieve the identifier of the first rule if there is any in the database. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the rule.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the rule. | |
| **Description:** | Retrieve the first Protocol-Based VLAN classification rule in the classifier database. When the user wants to retrieve all rules in the classifier, the user has to call this function at first and then call the ixVlanNextProtocolRuleIdGet for retrieving the consequent rules. This function will return the rule identifier of the first rule, instead of returning the content. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixVlanNextProtocolRuleIdGet** (RULE_ID ***rid***); | |
| **Parameters:** | **Description:** | **I/O:** |
| *rid | Address of the space used to retrieve the identifier of the rule in the database which is consequent to the previous one. The address cannot be NULL. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the rule.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the rule. | |
| **Description:** | Retrieve the Protocol-Based VLAN classification rule which is consequent to the rule be retrieved by calling this function or calling ixVlanFirstProtocolRuleIdGet at previous. This function will return the rule identifier instead of returning the rule content. | |

| Prototype: | IxVlanQosStatus **ixVlanProtocolRuleHitGet** (RULE_ID ***rid***, UINT32 ****hit***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| rid | The rule identifier specifying the rule whose hit count to be retrieved. | I |
| *hit | Address of the space used to retrieve the hi counter. The address cannot be NULL. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the hit counter.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the hit counter. | |
| **Description:** | Retrieve the hit counter of a Protocol-Based VLAN classification rule. The hit count records the times the rule is matched to the ingress or egress frames needs be classified in data path. | |

| Prototype: | IxVlanQosStatus **ixVlanProtocolRuleHitReset** (RULE_ID ***rid***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| rid | The rule identifier specifying the rule whose hit count to be reset. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on resetting the hit counter.<br>IX_VLAN_QOS_FAIL: Fail on resetting the hit counter. | |
| **Description:** | Reset the hit counter of a Protocol-Based VLAN classification rule to zero. | |

| Prototype: | IxVlanQosStatus **ixVlanProtocolRuleResetAll** (void); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| n/a | None | |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on resetting the Protocol-Based classification database.<br>IX_VLAN_QOS_FAIL: Fail on resetting the Protocol-Based classification database. | |
| **Description:** | Reset the database of the Protocol-Based classification. This action will remove all rules from the database. | |

| Prototype: | IxVlanQosStatus **ixVlanProtocolClassifierSet** (IxVlanQosPortId ***pid***, BOOL ***enabled***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | Identifier of the NPE Ethernet port to be set. | I |
| enabled | The Boolean value indicating the classifier to be enabled or disabled. TRUE value indicates enabling the classifier and FALSE value indicates disabling it. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on setting the classifier.<br>IX_VLAN_QOS_FAIL: Fail on setting the classifier. | |
| **Description:** | Enable or disable the Protocol-Based VLAN classification for the incoming or outgoing frames on a given NPE Ethernet port. | |

| **Prototype:** | IxVlanQosStatus **ixVlanProtocolClassifierGet** (IxVlanQosPortId ***pid***, BOOL ****enabled***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | Identifier of the NPE Ethernet port to retrieve from. | I |
| *enabled | Address of the space used to retrieve the port is whether enabled. Zero value indicates the port is disabled and nonzero value indicates the port is enabled. The address cannot be NULL. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the classifier.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the classifier. | |
| **Description:** | Retrieve the Boolean value from a given NPE Ethernet port which indicating whether the Protocol-Based VLAN classification on the port is enabled. | |

| **Prototype:** | IxVlanQosStatus **ixQosShaperEnable** (IxVlanQosPortId ***pid***, IxVlanQosDirection ***dir***, IxQosTcId ***tcid***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | Identifier of the NPE Ethernet port the shaper is standing on. | I |
| dir | Direction which the shaper is standing on. | I |
| tcid | Traffic class which the shaper is working for. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on enabling the shaper.<br>IX_VLAN_QOS_FAIL: Fail on enabling the shaper. | |
| **Description:** | Enable the shaper on a given NPE Ethernet port and a given traffic class at a given direction. As the shaper is enabled, the rate of traffic corresponding to the shaper will be limited on the behavior described by the shaper configuration. The shaper is disabled by default. | |

| **Prototype:** | IxVlanQosStatus **ixQosShaperDisable** (IxVlanQosPortId ***pid***, IxVlanQosDirection ***dir***, IxQosTcId ***tcid***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | Identifier of the NPE Ethernet port the shaper is standing on. | I |
| dir | Direction which the shaper is standing on. | I |
| tcid | Traffic class which the shaper is working for. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on disabling the shaper.<br>IX_VLAN_QOS_FAIL: Fail on disabling the shaper. | |
| **Description:** | Disable the shaper on a given NPE Ethernet port and a given traffic class at a given direction. As the shaper is disabled, the rate of traffic corresponding to the shaper will not be limited. | |

| Prototype: | IxVlanQosStatus i**xQosShaperEnabledGet** (IxVlanQosPortId **pid**, IxVlanQosDirection **dir**, IxQosTcId **tcid**, BOOL ***enabled**); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | Identifier of the NPE Ethernet port the shaper is standing on. | I |
| dir | Direction which the shaper is standing on. | I |
| tcid | Traffic class which the shaper is working for. | I |
| *enabled | Address of the space used to retrieve the Boolean value. Zero value indicates the port is disabled and nonzero value indicates the port is enabled. The address cannot be NULL. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the shaper status.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the shaper status. | |
| **Description:** | Return whether the shaper on a given NPE Ethernet port and a given traffic class at a given direction is enabled. | |

| Prototype: | IxVlanQosStatus **ixQosShaperRateSet** (IxVlanQosPortId ***pid***, IxVlanQosDirection ***dir***, IxQosTcId ***tcid***, UINT32 ***fps***, UINT32 ***bps***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | Identifier of the NPE Ethernet port the shaper is standing on. | I |
| dir | Direction which the shaper is standing on. | I |
| tcid | Traffic class which the shaper is working for. | I |
| fps | The average frame number per second (frame rate). | I |
| bps | The average bits number per second (bit rate). | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on enabling the shaper.<br>IX_VLAN_QOS_FAIL: Fail on enabling the shaper. | |
| **Description:** | Set the average rate for a given shaper. Average rate of the traffic class corresponding to the shaper will be limited to under below the given number. The rate can be specified in the unit of “frame number per second” or/and in the unit of “bit number per second”. If rates in both units were specified, the traffic class will be limited by which has been exceeded first. If the user wants to specify the frame rate to under a certain number and does not care the bit rate, for example, the user has to use IX_QOS_RATE_UNLIMIT to specify the bit rate. In the case that does not care the frame rate, use the same identifier for the parameter either. | |

| Prototype: | IxVlanQosStatus **ixQosShaperRateGet** (IxVlanQosPortId ***pid***, IxVlanQosDirection ***dir***, IxQosTcId ***tcid***, UINT32 ****fps***, UINT32 ****bps***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | Identifier of the NPE Ethernet port the shaper is standing on. | I |
| dir | Direction which the shaper is standing on. | I |
| tcid | Traffic class which the shaper is working for. | I |
| *fps | Address of the space used to retrieve the average frame rate. | O |
| *bps | Address of the space used to retrieve the average bit rate. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the rate.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the rate. | |
| **Description:** | Retrieve the average frame rate and average bit rate from a given shaper. | |

| **Prototype:** | IxVlanQosStatus **ixQosShaperCeilSet** (IxVlanQosPortId ***pid***, IxVlanQosDirection ***dir***, IxQosTcId ***tcid***, UINT32 ***fps***, UINT32 ***bps***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | Identifier of the NPE Ethernet port the shaper is standing on. | I |
| dir | Direction which the shaper is standing on. | I |
| tc | Traffic class which the shaper is working for. | I |
| fps | The ceil frame rate. | I |
| bps | The ceil bit rate. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on setting the ceil rate.<br>IX_VLAN_QOS_FAIL: Fail on setting the ceil rate. | |
| **Description:** | Set the ceil traffic rate for a given shaper. Ceil rate is the transient maximum rate the traffic was allowed to pass through the shaper. The time length that the traffic can be allowed to pass in the ceil rate is determined by how many traffic less than the average rate were accumulated before. For example, consider the condition that the average rate was set to 100 frames/sec and the ceil rate was set to 500 frames/sec for a shaper. If traffic rate was generated in 20 frames per second and was lasted for 10 seconds, there were 80 frames less than average rate per second and totally 800 frames were accumulated for the burst size during the 10 seconds. When the traffic suddenly increases to over 500 frames per second, the shaper can allow the traffic to pass through in the ceil rate, which is 400 frames larger than the average rate (over 400 frames were in burst per second), and it can last for 2 (800 / 400) seconds. The ceil rate can be specified in frame rate and/or bit rate. The traffic will be bounded by the rate which is exceeded at first. | |

| **Prototype:** | IxVlanQosStatus **ixQosShaperCeilGet** (IxVlanQosPortId ***pid***, IxVlanQosDirection ***dir***, IxQosTcId ***tcid***, UINT32 ****fps***, UINT32 ****bps***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | Identifier of the NPE Ethernet port the shaper is standing on. | I |
| dir | Direction which the shaper is standing on. | I |
| tcid | Traffic class which the shaper is working for. | I |
| *fps | Address of the space used to retrieve the ceil of frame rate. | O |
| *bps | Address of the space used to retrieve the ceil of frame rate. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the ceil rate.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the ceil rate. | |
| **Description:** | Retrieve the ceil frame rate and the ceil bit rate from a given shaper. | |

| Prototype: | IxVlanQosStatus **ixQosShaperBurstSizeSet** (IxVlanQosPortId ***pid***, IxVlanQosDirection ***dir***, IxQosTcId ***tcid***, UINT32 ***frames***, UINT32 ***bits***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | Identifier of the NPE Ethernet port the shaper is standing on. | I |
| dir | Direction which the shaper is standing on. | I |
| tcid | Traffic class which the shaper is working for. | I |
| frames | Amount of frames for burst. | I |
| bits | Amount of bits for burst. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on setting the burst size.<br>IX_VLAN_QOS_FAIL: Fail on setting the burst size. | |
| **Description:** | Set the burst size for a given shaper. The burst size means how many amount of traffic addition to the traffic under average rate can be allowed to pass through the shaper in the ceil rate. Refer to the example in the description for ixQosShaperCeilSet. | |

| Prototype: | IxVlanQosStatus **ixQosShaperBurstSizeGet** (IxVlanQosPortId ***pid***, IxVlanQosDirection ***dir***, IxQosTcId ***tcid***, UINT32 ****frames***, UINT32 ****bits***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | Identifier of the NPE Ethernet port the shaper is standing on. | I |
| dir | Direction which the shaper is standing on. | I |
| tcid | Traffic class which the shaper is working for. | I |
| *frames | Address of the space to retrieve the burst size for frames. | O |
| *bits | Address of the space to retrieve the burst size for bits. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the burst size.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the burst size. | |
| **Description:** | Retrieve the burst size of frames and bits for a given shaper. | |

| Prototype: | IxVlanQosStatus **ixQosPriorityMappingSet** (IxVlanQosPortId ***pid***, IxVlanQosDirection ***dir***, IxVlanQosPriority ***priority***, IxQosTcId ***tcid***); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | Identifier of the NPE Ethernet port the traffic class is standing on. | I |
| dir | Direction which the traffic class is standing on. | I |
| priority | The user priority to be mapped. | I |
| tcid | Traffic class which the user priority is mapped to. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on setting the traffic class.<br>IX_VLAN_QOS_FAIL: Fail on setting the traffic class. | |
| **Description:** | Set for which traffic class that a given user priority is mapped to on a given NPE Ethernet port and a given direction. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixQosPriorityMappingGet** (IxVlanQosPortId ***pid***, IxVlanQosDirection ***dir***, IxVlanQosPriority ***priority***, IxQosTcId ****tcid***); | |
| **Parameters:** | **Description:** | **I/O:** |
| pid | Identifier of the NPE Ethernet port the traffic class is standing on. | I |
| dir | Direction which the traffic class is standing on. | I |
| priority | The user priority to be mapped. | I |
| *tcid | Address of the space to retrieve the traffic which the user priority is mapped to. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the traffic class.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the traffic class. | |
| **Description:** | Retrieve the traffic class that a given user priority is mapped to on a given NPE Ethernet port and a given direction. | |

| | | |
|---|---|---|
| **Prototype:** | IxVlanQosStatus **ixQosPriorityMappingTableSet** (IxVlanQosPortId ***pid***, IxVlanQosDirection ***dir***, IxQosTcId ***tcid***[] ); | |
| **Parameters:** | **Description:** | **I/O:** |
| pid | Identifier of the NPE Ethernet port the mapping table is for. | I |
| dir | Direction that the mapping table is for. | I |
| tcid | Array of traffic classes for mapping the eight user priorities. The first element in the array specifies the traffic that user priority 0 is mapped to, and so on. | I |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on setting the table.<br>IX_VLAN_QOS_FAIL: Fail on setting the table. | |
| **Description:** | Set the table for specifying which traffic classes that eight user priorities are mapped to. | |

| Prototype: | IxVlanQosStatus **ixQosPriorityMappingTableGet** (IxVlanQosPortId ***pid***, IxVlanQosDirection ***dir***, IxQosTcId ***tcid***[] ); | |
|---|---|---|
| **Parameters:** | **Description:** | **I/O:** |
| pid | Identifier of the NPE Ethernet port the mapping table is from. | I |
| dir | Direction that the mapping table is from. | I |
| tcid | Array of traffic classes for mapping the eight user priorities. | O |
| **Return:** | IX_VLAN_QOS_SUCCESS: Success on retrieving the table.<br>IX_VLAN_QOS_FAIL: Fail on retrieving the table. | |
| **Description:** | Retrieve the table for specifying which traffic classes that eight user priorities are mapped to. | |

**This page is intentionally left blank.**