Workgroup: Internet Engineering Task Force
Internet-Draft:
draft-ietf-netmod-sub-intf-vlan-model-09
Published: 30 January 2024
Intended Status: Standards Track
Expires: 2 August 2024
Authors: R.G. Wilton, Ed. Cisco Systems
S. Mansfield, Ed. Ericsson

Sub-interface VLAN YANG Data Models

Abstract

This document defines YANG modules to add support for classifying traffic received on interfaces as Ethernet/VLAN framed packets to sub-interfaces based on the fields available in the Ethernet/VLAN frame headers. These modules allow configuration of Layer 3 and Layer 2 sub-interfaces (e.g. L2VPN attachment circuits) that can interoperate with IETF based forwarding protocols; such as IP and L3VPN services; or L2VPN services like VPWS, VPLS, and EVPN. The sub-interfaces also interoperate with VLAN tagged traffic orignating from an IEEE 802.1Q compliant bridge.

The model differs from an IEEE 802.1Q bridge model in that the configuration is interface/sub-interface based as opposed to being based on membership of an 802.1Q VLAN bridge.

The YANG data models in this document conforms to the Network Management Datastore Architecture (NMDA) defined in RFC 8342.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 2 August 2024.

Table of Contents

1. Introduction

This document defines two YANG [RFC7950] modules that augment the encapsulation choice YANG element defined in Interface Extensions

YANG [I-D.ietf-netmod-intf-ext-yang] and the generic interfaces data model defined in [RFC8343]. The two modules provide configuration nodes to support classification of Ethernet/VLAN traffic to sub-interfaces, that can have interface based feature and service configuration applied to them.

The purpose of these models is to allow IETF defined forwarding protocols, for example, IPv6 [RFC8200], Ethernet Pseudo Wires [RFC4448] and VPLS [RFC4761] [RFC4762], when configured via appropriate YANG data models [RFC8344] [I-D.ietf-bess-l2vpn-yang], to interoperate with VLAN tagged traffic received from an IEEE 802.1Q compliant bridge.

In the case of layer 2 Ethernet services, the flexible encapsulation module also supports flexible rewriting of the VLAN tags contained in the frame header.

For reference, a comparison between the sub-interface based YANG model documented in this draft and an IEEE 802.1Q bridge model is described in Appendix A.

In summary, the YANG modules defined in this internet draft are:

ietf-if-vlan-encapsulation.yang - Defines the model for basic classification of VLAN tagged traffic, normally to L3 packet forwarding services

ietf-if-flexible-encapsulation.yang - Defines the model for flexible classification of Ethernet/VLAN traffic, normally to L2 frame forwarding services

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The term 'sub-interface' is defined in section 2.6 of Interface Extensions YANG [I-D.ietf-netmod-intf-ext-yang].

1.2. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

1.3. Prefixes in Data Node Names

In this document, names of data nodes and other data model objects are prefixed using the standard prefix associated with the corresponding YANG imported modules, as shown in Table 1.

Prefix	YANG Module	Reference
if-vlan	ietf-if-vlan-encapsulation	This document
if-flex	ietf-if-flexible-encapsulation	This document
if-ext	ietf-if-extensions	[I-D.ietf-netmod-intf-ext-yang]
if	ietf-interfaces	[RFC8343]
ianaift	iana-if-type	[RFC7224]
dot1q-types	ieee802-dot1q-types	[IEEE 802.1Q 2022]

Table 1: Prefixes for imported YANG modules

2. Objectives

The primary aim of the YANG modules contained in this draft is to provide the core model that is required to implement VLAN transport services on router based devices that is fully compatible with IEEE 802.1Q compliant bridges.

A secondary aim is for the modules to be structured in such a way that they can be cleanly extended in future.

2.1. Interoperability with IEEE 802.1Q compliant bridges

The modules defined in this document are designed to fully interoperate with IEEE 802.1Q compliant bridges. In particular, the models are restricted to only matching, adding, or rewriting the 802.1Q VLAN tags in frames in ways that are compatible with IEEE 802.1Q compliant bridges.

3. Interface VLAN Encapsulation Model

The Interface VLAN encapsulation model provides appropriate leaves for termination of an 802.1Q VLAN tagged segment to a sub-interface (or interface) based L3 service, such as IP. It allows for termination of traffic with one or two 802.1Q VLAN tags.

The L3 service must be configured via a separate YANG data model, e.g., [RFC8344]. A short example of configuring 802.1Q VLAN sub-interfaces with IP using YANG is provided in Section 7.1.

The "ietf-if-vlan-encapsulation" YANG module has the following structure:

```
module: ietf-if-vlan-encapsulation
  augment /if:interfaces/if:interface/if-ext:encapsulation
            /if-ext:encaps-type:
    +--:(dot1q-vlan)
       +--rw dot1q-vlan
          +--rw outer-tag
          |  +--rw tag-type dot1q-tag-type
          |  +--rw vlan-id     vlanid
          +--rw second-tag!
             +--rw tag-type dot1q-tag-type
             +--rw vlan-id     vlanid
```

4. Interface Flexible Encapsulation Model

The Interface Flexible Encapsulation model is designed to allow for the flexible provisioning of layer 2 services. It provides the capability to classify and demultiplex Ethernet/VLAN frames received on an Ethernet trunk interface to sub-interfaces based on the fields available in the layer 2 headers. Once classified to sub-interfaces, it provides the capability to selectively modify fields within the layer 2 frame header before the frame is handed off to the appropriate forwarding code for further handling. The forwarding instance, e.g., L2VPN, VPLS, etc., is configured using a separate YANG configuration model defined elsewhere, e.g., [I-D.ietf-bess-l2vpn-yang].

The model supports a common core set of layer 2 header matches based on the 802.1Q tag type and VLAN Ids contained within the header up to a tag stack depth of two tags.

The model supports flexible rewrites of the layer 2 frame header for data frames as they are processed on the interface. It defines a set of standard tag manipulations that allow for the insertion, removal, or rewrite of one or two 802.1Q VLAN tags. The expectation is that manipulations are generally implemented in a symmetrical fashion, i.e. if a manipulation is performed on ingress traffic on an interface then the reverse manipulation is always performed on egress traffic out of the same interface. However, the model also allows for asymmetrical rewrites, which may be required to implement some forwarding models (such as E-Tree).

The model also allows a flexible encapsulation and rewrite to be configured directly on an Ethernet or LAG interface without configuring separate child sub-interfaces. Ingress frames that do

not match the encapsulation are dropped. Egress frames MUST conform to the encapsulation.

The final aim for the model design is for it to be cleanly extensible to add in additional match and rewrite criteria of the layer 2 header, such as matching on the source or destination MAC address, PCP or DEI fields in the 802.1Q tags, or the EtherType of the frame payload. Rewrites can also be extended to allow for modification of other fields within the layer 2 frame header.

A short example of configuring 802.1Q VLAN sub-interfaces with L2VPN using YANG is provided in Section 7.2.

The "ietf-if-flexible-encapsulation" YANG module has the following structure:

```
module: ietf-if-flexible-encapsulation
  augment /if:interfaces/if:interface/if-ext:encapsulation
            /if-ext:encaps-type:
    +--:(flexible)
       +--rw flexible
          +--rw match
          |  +--rw (match-type)
          |     +--:(default)
          |     |  +--rw default?                empty
          |     +--:(untagged)
          |     |  +--rw untagged?               empty
          |     +--:(dot1q-priority-tagged)
          |     |  +--rw dot1q-priority-tagged
          |     |     +--rw tag-type dot1q-types:dot1q-tag-type
          |     +--:(dot1q-vlan-tagged)
          |        +--rw dot1q-vlan-tagged
          |           +--rw outer-tag
          |           |  +--rw tag-type dot1q-tag-type
          |           |  +--rw vlan-id     union
          |           +--rw second-tag!
          |           |  +--rw tag-type dot1q-tag-type
          |           |  +--rw vlan-id     union
          |           +--rw match-exact-tags?   empty
          +--rw rewrite {flexible-rewrites}?
          |  +--rw (direction)?
          |     +--:(symmetrical)
          |     |  +--rw symmetrical
          |     |     +--rw dot1q-tag-rewrite {dot1q-tag-rewrites}?
          |     |        +--rw pop-tags?    uint8
          |     |        +--rw push-tags!
          |     |           +--rw outer-tag
          |     |           |  +--rw tag-type dot1q-tag-type
          |     |           |  +--rw vlan-id     vlanid
          |     |           +--rw second-tag!
          |     |              +--rw tag-type dot1q-tag-type
          |     |              +--rw vlan-id     vlanid
          |     +--:(asymmetrical) {asymmetric-rewrites}?
          |        +--rw ingress
          |        |  +--rw dot1q-tag-rewrite {dot1q-tag-rewrites}?
          |        |     +--rw pop-tags?    uint8
          |        |     +--rw push-tags!
          |        |        +--rw outer-tag
          |        |        |  +--rw tag-type dot1q-tag-type
          |        |        |  +--rw vlan-id     vlanid
          |        |        +--rw second-tag!
          |        |           +--rw tag-type dot1q-tag-type
          |        |           +--rw vlan-id     vlanid
          |        +--rw egress
```

```
|           +--rw dot1q-tag-rewrite {dot1q-tag-rewrites}?
|              +--rw pop-tags?    uint8
|              +--rw push-tags!
|                 +--rw outer-tag
|                 |  +--rw tag-type dot1q-tag-type
|                 |  +--rw vlan-id     vlanid
|                 +--rw second-tag!
|                    +--rw tag-type dot1q-tag-type
|                    +--rw vlan-id     vlanid
+--rw local-traffic-default-encaps!
   +--rw outer-tag
   |  +--rw tag-type dot1q-tag-type
   |  +--rw vlan-id     vlanid
   +--rw second-tag!
      +--rw tag-type dot1q-tag-type
      +--rw vlan-id     vlanid
```

5. VLAN Encapsulation YANG Module

This YANG module augments the 'encapsulation' container defined in ietf-if-extensions.yang [I-D.ietf-netmod-intf-ext-yang]. It also contains references to [RFC8343], [RFC7224], and [IEEE 802.1Q 2022].

```
<CODE BEGINS> file "ietf-if-vlan-encapsulation@2023-01-26.yang"

module ietf-if-vlan-encapsulation {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation";
  prefix if-vlan;

  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model For Interface Management";
  }

  import iana-if-type {
    prefix ianaift;
    reference
      "RFC 7224: IANA Interface Type YANG Module";
  }

  import ieee802-dot1q-types {
    prefix dot1q-types;
    revision-date 2022-01-19;
    reference
      "IEEE Std 802.1Q-2022: IEEE Standard for Local and
       metropolitan area networks -- Bridges and Bridged Networks";
  }

  import ietf-if-extensions {
    prefix if-ext;
    reference
      "RFC XXXX: Common Interface Extension YANG Data Models";
  }

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

  contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     Editor:   Robert Wilton
               <mailto:rwilton@cisco.com>";

  description
    "This YANG module models configuration to classify IEEE 802.1Q
     VLAN tagged Ethernet traffic by exactly matching the tag type
     and VLAN identifier of one or two 802.1Q VLAN tags in the frame.

     Copyright (c) 2023 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.
```

```
   Redistribution and use in source and binary forms, with or
   without modification, is permitted pursuant to, and subject to
   the license terms contained in, the Revised BSD License set
   forth in Section 4.c of the IETF Trust's Legal Provisions
   Relating to IETF Documents
   (https://trustee.ietf.org/license-info).

   This version of this YANG module is part of RFC XXXX
   (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
   for full legal notices.

   The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
   NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
   'MAY', and 'OPTIONAL' in this document are to be interpreted as
   described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
   they appear in all capitals, as shown here.";

 revision 2023-01-26 {
   description
     "Latest draft revision";
   reference
     "RFC XXXX: Sub-interface VLAN YANG Data Models";
 }

 augment "/if:interfaces/if:interface/if-ext:encapsulation/"
       + "if-ext:encaps-type" {
   when "derived-from-or-self(../if:type,
                              'ianaift:ethernetCsmacd') or
         derived-from-or-self(../if:type,
                              'ianaift:ieee8023adLag') or
         derived-from-or-self(../if:type, 'ianaift:l2vlan') or
         derived-from-or-self(../if:type,
                              'if-ext:ethSubInterface')" {
       description
         "Applies only to Ethernet-like interfaces and
          sub-interfaces.";
   }

   description
     "Augment the generic interface encapsulation with basic 802.1Q
      VLAN tag classifications";

   case dot1q-vlan {
     container dot1q-vlan {

       description
         "Classifies 802.1Q VLAN tagged Ethernet frames to a
          sub-interface (or interface) by exactly matching the
          number of tags, tag type(s) and VLAN identifier(s).
```

```
   Only frames matching the classification configured on a
   sub-interface/interface are processed on that
   sub-interface/interface.

   Frames that do not match any sub-interface are processed
   directly on the parent interface, if it is associated with
   a forwarding instance, otherwise they are dropped.";

container outer-tag {
  must 'tag-type = "dot1q-types:s-vlan" or '
     + 'tag-type = "dot1q-types:c-vlan"' {

    error-message
      "Only C-VLAN and S-VLAN tags can be matched.";

    description
      "For IEEE 802.1Q interoperability, only C-VLAN and
       S-VLAN tags are matched.";
  }

  description
    "Specifies the VLAN tag values to match against the
     outermost (first) 802.1Q VLAN tag in the frame.";

  uses dot1q-types:dot1q-tag-classifier-grouping;
}

container second-tag {
  must '../outer-tag/tag-type = "dot1q-types:s-vlan" and '
     + 'tag-type = "dot1q-types:c-vlan"' {

    error-message
      "When matching two 802.1Q VLAN tags, the outermost
       (first) tag in the frame MUST be specified and be of
       S-VLAN type and the second tag in the frame must be of
       C-VLAN tag type.";

    description
      "For IEEE 802.1Q interoperability, when matching two
       802.1Q VLAN tags, it is REQUIRED that the outermost
       tag exists and is an S-VLAN, and the second tag is a
       C-VLAN.";
  }

  presence "Classify frames that have two 802.1Q VLAN tags.";

  description
    "Specifies the VLAN tag values to match against the
     second outermost 802.1Q VLAN tag in the frame.";
```

```
          uses dot1q-types:dot1q-tag-classifier-grouping;
        }
      }
    }
  }
}

<CODE ENDS>
```

6. Flexible Encapsulation YANG Module

This YANG module augments the 'encapsulation' container defined in ietf-if-extensions.yang [I-D.ietf-netmod-intf-ext-yang]. This YANG module also augments the 'interface' list entry defined in [RFC8343]. It also contains references to [RFC7224], and [IEEE_802.1Q_2022].

```
<CODE BEGINS> file "ietf-if-flexible-encapsulation@2023-01-26.yang"

module ietf-if-flexible-encapsulation {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-if-flexible-encapsulation";
  prefix if-flex;

  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model For Interface Management";
  }

  import iana-if-type {
    prefix ianaift;
    reference
      "RFC 7224: IANA Interface Type YANG Module";
  }

  import ieee802-dot1q-types {
    prefix dot1q-types;
    revision-date 2022-01-19;
    reference
      "IEEE Std 802.1Q-2022: IEEE Standard for Local and
       metropolitan area networks -- Bridges and Bridged Networks";
  }

  import ietf-if-extensions {
    prefix if-ext;
    reference
      "RFC XXXX: Common Interface Extension YANG Data Models";
  }

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

  contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     Editor:   Robert Wilton
               <mailto:rwilton@cisco.com>";

  description
    "This YANG module describes interface configuration for flexible
     classification and rewrites of IEEE 802.1Q VLAN tagged Ethernet
     traffic.

     Copyright (c) 2022 IETF Trust and the persons identified as
```

```
   authors of the code.  All rights reserved.

   Redistribution and use in source and binary forms, with or
   without modification, is permitted pursuant to, and subject to
   the license terms contained in, the Revised BSD License set
   forth in Section 4.c of the IETF Trust's Legal Provisions
   Relating to IETF Documents
   (https://trustee.ietf.org/license-info).

   This version of this YANG module is part of RFC XXXX
   (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
   for full legal notices.

   The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
   NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
   'MAY', and 'OPTIONAL' in this document are to be interpreted as
   described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
   they appear in all capitals, as shown here.";

revision 2023-01-26 {
  description
    "Latest draft revision";
  reference
    "RFC XXXX: Sub-interface VLAN YANG Data Models";
}

feature flexible-rewrites {
  description
    "This feature indicates that the network element supports
      specifying flexible rewrite operations.";
}

feature asymmetric-rewrites {
  description
    "This feature indicates that the network element supports
     specifying different rewrite operations for the ingress
     rewrite operation and egress rewrite operation.";
}

feature dot1q-tag-rewrites {
  description
    "This feature indicates that the network element supports the
     flexible rewrite functionality specifying 802.1Q tag
     rewrites.";
}

grouping flexible-match {
  description
    "Represents a flexible frame classification:
```

```
   The rules for a flexible match are:
     1. Match-type: default, untagged, priority tag, or tag
        stack.
     2. Each tag in the stack of tags matches:
      a. tag type (802.1Q or 802.1ad) +
      b. tag value:
        i.   single tag
        ii.  set of tag ranges/values.
        iii. 'any' keyword";

choice match-type {
  mandatory true;

  description
    "Provides a choice of how the frames may be
     matched";

  case default {
    description
      "Default match";

    leaf default {
      type empty;

      description
        "Default match.  Matches all traffic not matched to any
         other peer sub-interface by a more specific
         encapsulation.";
    }
  }

  case untagged {
    description
      "Match untagged Ethernet frames only";

    leaf untagged {
      type empty;

      description
        "Untagged match.  Matches all untagged traffic.";
    }
  }

  case dot1q-priority-tagged {
    description
      "Match 802.1Q priority tagged Ethernet frames only";

    container dot1q-priority-tagged {
      description
        "802.1Q priority tag match";
```

```
    leaf tag-type {
      type dot1q-types:dot1q-tag-type;
      mandatory true;

      description
        "The 802.1Q tag type of matched priority
         tagged packets";
    }
  }
}

case dot1q-vlan-tagged {
  container dot1q-vlan-tagged {
    description
      "Matches VLAN tagged frames";

    container outer-tag {
      must 'tag-type = "dot1q-types:s-vlan" or '
         + 'tag-type = "dot1q-types:c-vlan"' {

        error-message
          "Only C-VLAN and S-VLAN tags can be matched.";

        description
          "For IEEE 802.1Q interoperability, only C-VLAN and
           S-VLAN tags can be matched.";
      }

      description
        "Classifies traffic using the outermost (first) VLAN
         tag on the frame.";

      uses "dot1q-types:"
         + "dot1q-tag-ranges-or-any-classifier-grouping";
    }

    container second-tag {
      must
        '../outer-tag/tag-type = "dot1q-types:s-vlan" and '
      + 'tag-type = "dot1q-types:c-vlan"' {

        error-message
          "When matching two tags, the outermost (first) tag
           must be specified and of S-VLAN type and the second
           outermost tag must be of C-VLAN tag type.";

        description
          "For IEEE 802.1Q interoperability, when matching two
           tags, it is required that the outermost (first) tag
```

```
              exists and is an S-VLAN, and the second outermost
              tag is a C-VLAN.";
          }

          presence "Also classify on the second VLAN tag.";

          description
            "Classifies traffic using the second outermost VLAN tag
             on the frame.";

          uses "dot1q-types:"
             + "dot1q-tag-ranges-or-any-classifier-grouping";
        }

        leaf match-exact-tags {
          type empty;
          description
            "If set, indicates that all 802.1Q VLAN tags in the
             Ethernet frame header must be explicitly matched, i.e.
             the EtherType following the matched tags must not be a
             802.1Q tag EtherType.  If unset then extra 802.1Q VLAN
             tags are allowed.";
        }
      }
    }
  }
}

grouping dot1q-tag-rewrite {
  description
    "Flexible rewrite grouping.  Can be either be expressed
     symmetrically, or independently in the ingress and/or egress
     directions.";

  leaf pop-tags {
    type uint8 {
      range "1..2";
    }

    description
      "The number of 802.1Q VLAN tags to pop, or translate if used
       in conjunction with push-tags.

       Popped tags are the outermost tags on the frame.";
  }

  container push-tags {
    presence "802.1Q tags are pushed or translated";

    description
```

```
      "The 802.1Q tags to push on the front of the frame, or
       translate if configured in conjunction with pop-tags.";

    container outer-tag {
      must 'tag-type = "dot1q-types:s-vlan" or '
         + 'tag-type = "dot1q-types:c-vlan"' {

        error-message "Only C-VLAN and S-VLAN tags can be pushed.";

        description
          "For IEEE 802.1Q interoperability, only C-VLAN and S-VLAN
           tags can be pushed.";
      }

      description
        "The outermost (first) VLAN tag to push onto the frame.";

      uses dot1q-types:dot1q-tag-classifier-grouping;
    }

    container second-tag {
      must '../outer-tag/tag-type = "dot1q-types:s-vlan" and '
         + 'tag-type = "dot1q-types:c-vlan"' {

        error-message
          "When pushing/rewriting two tags, the outermost tag must
           be specified and of S-VLAN type and the second outermost
           tag must be of C-VLAN tag type.";

        description
          "For IEEE 802.1Q interoperability, when pushing two tags,
           it is required that the outermost tag exists and is an
           S-VLAN, and the second outermost tag is a C-VLAN.";
      }

      presence
        "In addition to the first tag, also push/rewrite a second
         VLAN tag.";

      description
        "The second outermost VLAN tag to push onto the frame.";

      uses dot1q-types:dot1q-tag-classifier-grouping;
    }
  }
}

grouping flexible-rewrite {
  description
    "Grouping for flexible rewrites of fields in the L2 header.
```

```
     Restricted to flexible 802.1Q VLAN tag rewrites, but could be
     extended to cover rewrites of other fields in the L2 header in
     future.";

  container dot1q-tag-rewrite {
    if-feature "dot1q-tag-rewrites";

    description
      "802.1Q VLAN tag rewrite.

       Translate operations are expressed as a combination of tag
       push and pop operations.  E.g., translating the outer tag is
       expressed as popping a single tag, and pushing a single tag.
       802.1Q tags that are translated SHOULD preserve the PCP and
       DEI fields unless if a different QoS behavior has been
       specified.";
    uses dot1q-tag-rewrite;
  }
}

augment "/if:interfaces/if:interface/if-ext:encapsulation/"
      + "if-ext:encaps-type" {
  when "derived-from-or-self(../if:type,
                             'ianaift:ethernetCsmacd') or
        derived-from-or-self(../if:type,
                             'ianaift:ieee8023adLag') or
        derived-from-or-self(../if:type, 'ianaift:l2vlan') or
        derived-from-or-self(../if:type,
                             'if-ext:ethSubInterface')" {

      description
        "Applies only to Ethernet-like interfaces and
         sub-interfaces.";
  }

  description
    "Augment the generic interface encapsulation with flexible
     match and rewrite for VLAN sub-interfaces.";

  case flexible {
    description
      "Flexible encapsulation and rewrite";

    container flexible {
      description
        "Flexible encapsulation allows for the matching of ranges
         and sets of 802.1Q VLAN Tags and performing rewrite
         operations on the VLAN tags.
```

```
   The structure is also designed to be extended to allow for
   matching/rewriting other fields within the L2 frame header
   if required.";

container match {
  description
    "Flexibly classifies Ethernet frames to a sub-interface
     (or interface) based on the L2 header fields.

     Only frames matching the classification configured on a
     sub-interface/interface are processed on that
     sub-interface/interface.

     Frames that do not match any sub-interface are processed
     directly on the parent interface, if it is associated
     with a forwarding instance, otherwise they are dropped.

     If a frame could be classified to multiple
     sub-interfaces then they get classified to the
     sub-interface with the most specific match.  E.g.,
     matching two VLAN tags in the frame is more specific
     than matching the outermost VLAN tag, which is more
     specific than the catch all 'default' match.";

  uses flexible-match;
}

container rewrite {
  if-feature "flexible-rewrites";

  description
    "L2 frame rewrite operations.

     Rewrites allows for modifications to the L2 frame header
     as it transits the interface/sub-interface.  Examples
     include adding a VLAN tag, removing a VLAN tag, or
     rewriting the VLAN Id carried in a VLAN tag.";

  choice direction {
    description
      "Whether the rewrite policy is symmetrical or
       asymmetrical.";

    case symmetrical {
      container symmetrical {
        uses flexible-rewrite;

        description
          "Symmetrical rewrite.  Expressed in the ingress
           direction, but the reverse operation is applied to
```

```
            egress traffic.

            E.g., if a tag is pushed on ingress traffic, then
            the reverse operation is a 'pop 1', that is
            performed on traffic egressing the interface, so
            a peer device sees a consistent L2 encapsulation
            for both ingress and egress traffic.";
        }
      }

      case asymmetrical {
        if-feature "asymmetric-rewrites";

        description
          "Asymmetrical rewrite.

           Rewrite operations may be specified in only a single
           direction, or different rewrite operations may be
           specified in each direction.";

        container ingress {
          uses flexible-rewrite;

          description
            "A rewrite operation that only applies to ingress
             traffic.

             Ingress rewrite operations are performed before
             the frame is subsequently processed by the
             forwarding operation.";
        }

        container egress {
          uses flexible-rewrite;

          description
            "A rewrite operation that only applies to egress
             traffic.";
        }
      }
    }
  }

  container local-traffic-default-encaps {
    presence "A local traffic default encapsulation has been
              specified.";

    description
      "Specifies the 802.1Q VLAN tags to use by default for
       locally sourced traffic from the interface.
```

```
   Used for encapsulations that match a range of VLANs (or
   'any'), where the source VLAN Ids are otherwise
   ambiguous.";

container outer-tag {
  must 'tag-type = "dot1q-types:s-vlan" or '
     + 'tag-type = "dot1q-types:c-vlan"' {

    error-message
      "Only C-VLAN and S-VLAN tags can be matched.";

    description
      "For IEEE 802.1Q interoperability, only C-VLAN and
       S-VLAN tags can be matched.";
  }

  description
    "The outermost (first) VLAN tag for locally sourced
     traffic.";

  uses dot1q-types:dot1q-tag-classifier-grouping;
}

container second-tag {
  must
    '../outer-tag/tag-type = "dot1q-types:s-vlan" and '
  + 'tag-type = "dot1q-types:c-vlan"' {

    error-message
      "When specifying two tags, the outermost (first) tag
       must be specified and of S-VLAN type and the second
       outermost tag must be of C-VLAN tag type.";

    description
      "For IEEE 802.1Q interoperability, when specifying
       two tags, it is required that the outermost (first)
       tag exists and is an S-VLAN, and the second
       outermost tag is a C-VLAN.";
  }

  presence
    "Indicates existence of a second outermost VLAN tag.";

  description
    "The second outermost VLAN tag for locally sourced
     traffic.";

  uses dot1q-types:dot1q-tag-classifier-grouping;
}
```

```
        }
      }
    }
  }
}

<CODE ENDS>
```

7. Examples

The following sections give examples of configuring a sub-interface supporting L3 forwarding, and a sub-interface being used in conjunction with the IETF L2VPN YANG model [I-D.ietf-bess-l2vpn-yang].

7.1. Layer 3 sub-interfaces with IPv6

This example illustrates two layer sub-interfaces, 'eth0.1' and 'eth0.2', both are child interfaces of the Ethernet interface 'eth0'.

'eth0.1' is configured to match traffic with two VLAN tags: an outer S-VLAN of 10 and an inner C-VLAN of 20.

'eth0.2' is configured to match traffic with a single S-VLAN tag, with VLAN Id 11.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces
  xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
  xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
  xmlns:dot1q-types="urn:ieee:std:802.1Q:yang:ieee802-dot1q-types"
  xmlns:if-ext="urn:ietf:params:xml:ns:yang:ietf-if-extensions">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
    </interface>
    <interface>
      <name>eth0.1</name>
      <type>ianaift:l2vlan</type>
      <if-ext:parent-interface>eth0</if-ext:parent-interface>
      <if-ext:encapsulation>
        <dot1q-vlan
         xmlns=
            "urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation">
          <outer-tag>
            <tag-type>dot1q-types:s-vlan</tag-type>
            <vlan-id>10</vlan-id>
          </outer-tag>
          <second-tag>
            <tag-type>dot1q-types:c-vlan</tag-type>
            <vlan-id>20</vlan-id>
          </second-tag>
        </dot1q-vlan>
      </if-ext:encapsulation>
      <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
        <forwarding>true</forwarding>
        <address>
          <ip>2001:db8:10::1</ip>
          <prefix-length>48</prefix-length>
        </address>
      </ipv6>
    </interface>
    <interface>
      <name>eth0.2</name>
      <type>ianaift:l2vlan</type>
      <if-ext:parent-interface>eth0</if-ext:parent-interface>
      <if-ext:encapsulation>
        <dot1q-vlan
         xmlns=
            "urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation">
          <outer-tag>
            <tag-type>dot1q-types:s-vlan</tag-type>
            <vlan-id>11</vlan-id>
```

```
          </outer-tag>
        </dot1q-vlan>
      </if-ext:encapsulation>
      <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
        <forwarding>true</forwarding>
        <address>
          <ip>2001:db8:11::1</ip>
          <prefix-length>48</prefix-length>
        </address>
      </ipv6>
    </interface>
  </interfaces>
</config>
```

7.2. Layer 2 sub-interfaces with L2VPN

This example illustrates a layer 2 sub-interface 'eth0.3' configured to match traffic with a S-VLAN tag of 10, and C-VLAN tag of 21; and remov the outer tag (S-VLAN 10) before the traffic is passed off to the L2VPN service.

It also illustrates another sub-interface 'eth1.0' under a separate physical interface configured to match traffic with a C-VLAN of 50, with the tag removed before traffic is given to any service. Sub-interface 'eth1.0' is not currently bound to any service and hence traffic classified to that sub-interface is dropped.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces
  xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
  xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
  xmlns:dot1q-types="urn:ieee:std:802.1Q:yang:ieee802-dot1q-types"
  xmlns:if-ext="urn:ietf:params:xml:ns:yang:ietf-if-extensions">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
    </interface>
    <interface>
      <name>eth0.3</name>
      <type>ianaift:l2vlan</type>
      <if-ext:parent-interface>eth0</if-ext:parent-interface>
      <if-ext:encapsulation>
        <flexible xmlns=
        "urn:ietf:params:xml:ns:yang:ietf-if-flexible-encapsulation">
          <match>
            <dot1q-vlan-tagged>
              <outer-tag>
                <tag-type>dot1q-types:s-vlan</tag-type>
                <vlan-id>10</vlan-id>
              </outer-tag>
              <second-tag>
                <tag-type>dot1q-types:c-vlan</tag-type>
                <vlan-id>21</vlan-id>
              </second-tag>
            </dot1q-vlan-tagged>
          </match>
          <rewrite>
            <symmetrical>
              <dot1q-tag-rewrite>
                <pop-tags>1</pop-tags>
              </dot1q-tag-rewrite>
            </symmetrical>
          </rewrite>
        </flexible>
      </if-ext:encapsulation>
    </interface>
    <interface>
      <name>eth1</name>
      <type>ianaift:ethernetCsmacd</type>
    </interface>
    <interface>
      <name>eth1.0</name>
      <type>ianaift:l2vlan</type>
      <if-ext:parent-interface>eth0</if-ext:parent-interface>
```

```
    <if-ext:encapsulation>
      <flexible xmlns=
      "urn:ietf:params:xml:ns:yang:ietf-if-flexible-encapsulation">
        <match>
          <dot1q-vlan-tagged>
            <outer-tag>
              <tag-type>dot1q-types:c-vlan</tag-type>
              <vlan-id>50</vlan-id>
            </outer-tag>
          </dot1q-vlan-tagged>
        </match>
        <rewrite>
          <symmetrical>
            <dot1q-tag-rewrite>
              <pop-tags>1</pop-tags>
            </dot1q-tag-rewrite>
          </symmetrical>
        </rewrite>
      </flexible>
    </if-ext:encapsulation>
  </interface>
</interfaces>
<network-instances
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-instance">
  <network-instance
   xmlns:l2vpn="urn:ietf:params:xml:ns:yang:ietf-l2vpn">
    <name>p2p-l2-1</name>
    <description>Point to point L2 service</description>
    <l2vpn:type>l2vpn:vpws-instance-type</l2vpn:type>
    <l2vpn:signaling-type>
      l2vpn:ldp-signaling
    </l2vpn:signaling-type>
    <endpoint xmlns="urn:ietf:params:xml:ns:yang:ietf-l2vpn">
      <name>local</name>
      <ac>
        <name>eth0.3</name>
      </ac>
    </endpoint>
    <endpoint xmlns="urn:ietf:params:xml:ns:yang:ietf-l2vpn">
      <name>remote</name>
      <pw>
        <name>pw1</name>
      </pw>
    </endpoint>
    <vsi-root>
    <!-- Does not Validate -->
    </vsi-root>
  </network-instance>
</network-instances>
```

```
  <pseudowires
      xmlns="urn:ietf:params:xml:ns:yang:ietf-pseudowires">
    <pseudowire>
      <name>pw1</name>
        <peer-ip>2001:db8::50></peer-ip>
        <pw-id>100</pw-id>
    </pseudowire>
  </pseudowires>
</config>
```

8. Acknowledgements

The authors would particularly like to thank Benoit Claise, John Messenger, Glenn Parsons, and Dan Romascanu for their help progressing this draft.

The authors would also like to thank Martin Bjorklund, Alex Campbell, Don Fedyk, Eric Gray, Giles Heron, Marc Holness, Iftekhar Hussain, Neil Ketley, William Lupton, John Messenger, Glenn Parsons, Ludwig Pauwels, Joseph White, Vladimir Vassilev, and members of the IEEE 802.1 WG for their helpful reviews and feedback on this draft.

9. IANA Considerations

9.1. YANG Module Registrations

The following YANG modules are requested to be registered in the IANA "YANG Module Names" [RFC6020] registry:

The ietf-if-vlan-encapsulation module:

Name: ietf-if-vlan-encapsulation

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation

Prefix: if-vlan

Reference: RFCXXXX

The ietf-if-flexible-encapsulation module:

Name: ietf-if-flexible-encapsulation

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-if-flexible-encapsulation

Prefix: if-flex

Reference: RFCXXXX

This document registers two URIs in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registrations have been made.

URI: urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-if-flexible-encapsulation

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

10. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory to implement secure transport is SSH [RFC6242] The NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of data nodes defined in this YANG module which are writable/creatable/deletable (i.e. config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g. edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

10.1. ietf-if-vlan-encapsulation.yang

The nodes in the vlan encapsulation YANG module are concerned with matching particular frames received on the network device to connect them to a layer 3 forwarding instance, and as such adding/modifying/deleting these nodes has a high risk of causing traffic to be lost because it is not being classified correctly, or is being classified to a separate sub-interface. The nodes, all under the subtree / interfaces/interface/encapsulation/dot1q-vlan, that are sensitive to this are:

*outer-tag/tag-type

*outer-tag/vlan-id

*second-tag/tag-type

*second-tag/vlan-id

10.2. ietf-if-flexible-encapsulation.yang

There are many nodes in the flexible encapsulation YANG module that are concerned with matching particular frames received on the network device, and as such adding/modifying/deleting these nodes

has a high risk of causing traffic to be lost because it is not being classified correctly, or is being classified to a separate sub-interface. The nodes, all under the subtree /interfaces/interface/encapsulation/flexible/match, that are sensitive to this are:

* *default
* *untagged
* *dot1q-priority-tagged
* *dot1q-priority-tagged/tag-type
* *dot1q-vlan-tagged/outer-tag/vlan-type
* *dot1q-vlan-tagged/outer-tag/vlan-id
* *dot1q-vlan-tagged/second-tag/vlan-type
* *dot1q-vlan-tagged/second-tag/vlan-id

There are also many modes in the flexible encapsulation YANG module that are concerned with rewriting the fields in the L2 header for particular frames received on the network device, and as such adding/modifying/deleting these nodes has a high risk of causing traffic to be dropped or incorrectly processed on peer network devices, or it could cause layer 2 tunnels to go down due to a mismatch in negotiated MTU. The nodes, all under the subtree /interfaces/interface/encapsulation/flexible/rewrite, that are sensitive to this are:

* *symmetrical/dot1q-tag-rewrite/pop-tags
* *symmetrical/dot1q-tag-rewrite/push-tags/outer-tag/tag-type
* *symmetrical/dot1q-tag-rewrite/push-tags/outer-tag/vlan-id
* *symmetrical/dot1q-tag-rewrite/push-tags/second-tag/tag-type
* *symmetrical/dot1q-tag-rewrite/push-tags/second-tag/vlan-id
* *asymmetrical/ingress/dot1q-tag-rewrite/pop-tags
* *asymmetrical/ingress/dot1q-tag-rewrite/push-tags/outer-tag/tag-type
* *asymmetrical/ingress/dot1q-tag-rewrite/push-tags/outer-tag/vlan-id

*asymmetrical/ingress/dot1q-tag-rewrite/push-tags/second-tag/tag-type

*asymmetrical/ingress/dot1q-tag-rewrite/push-tags/second-tag/vlan-id

*asymmetrical/egress/dot1q-tag-rewrite/pop-tags

*asymmetrical/egress/dot1q-tag-rewrite/push-tags/outer-tag/tag-type

*asymmetrical/egress/dot1q-tag-rewrite/push-tags/outer-tag/vlan-id

*asymmetrical/egress/dot1q-tag-rewrite/push-tags/second-tag/tag-type

*asymmetrical/egress/dot1q-tag-rewrite/push-tags/second-tag/vlan-id

Nodes in the flexible-encapsulation YANG module that are concerned with the VLAN tags to use for traffic sourced from the network element could cause protocol sessions (such as CFM) to fail if they are added, modified or deleted. The nodes, all under the subtree /interfaces/interface/flexible-encapsulation/local-traffic-default-encaps that are sensitive to this are:

*outer-tag/vlan-type

*outer-tag/vlan-id

*second-tag/vlan-type

*second-tag/vlan-id

11. References

11.1. Normative References

[I-D.ietf-netmod-intf-ext-yang] Wilton, R. and S. Mansfield, "Common Interface Extension YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-intf-ext-yang-12, 18 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-intf-ext-yang-12>.

[IEEE_802.1Q_2022] IEEE, "IEEE Standard for Local and Metropolitan Area Networks--Bridges and Bridged Networks", IEEE 802-1q-2022, DOI 10.1109/IEEESTD.2022.10004498, 30 December 2022, <https://ieeexplore.ieee.org/document/10004498>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", RFC 7224, DOI 10.17487/RFC7224, May 2014, <https://www.rfc-editor.org/info/rfc7224>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC8344] Bjorklund, M., "A YANG Data Model for IP Management", RFC 8344, DOI 10.17487/RFC8344, March 2018, <https://www.rfc-editor.org/info/rfc8344>.

11.2. Informative References

[I-D.ietf-bess-l2vpn-yang] Shah, H. C., Brissette, P., Chen, I., Hussain, I., Wen, B., and K. Tiruveedhula, "YANG Data Model for MPLS-based L2VPN", Work in Progress, Internet-Draft, draft-ietf-bess-l2vpn-yang-10, 2 July 2019, <https://datatracker.ietf.org/doc/html/draft-ietf-bess-l2vpn-yang-10>.

[RFC4448] Martini, L., Ed., Rosen, E., El-Aawar, N., and G. Heron, "Encapsulation Methods for Transport of Ethernet over

MPLS Networks", RFC 4448, DOI 10.17487/RFC4448, April 2006, <https://www.rfc-editor.org/info/rfc4448>.

[RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <https://www.rfc-editor.org/info/rfc4761>.

[RFC4762] Lasserre, M., Ed. and V. Kompella, Ed., "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling", RFC 4762, DOI 10.17487/RFC4762, January 2007, <https://www.rfc-editor.org/info/rfc4762>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

Appendix A. Comparison with the IEEE 802.1Q Configuration Model

In addition to the sub-interface based YANG model proposed here, the IEEE 802.1Q working group has developed a YANG model for the configuration of 802.1Q VLANs. This raises the valid question as to whether the models overlap and whether it is necessary or beneficial to have two different models for superficially similar constructs. This section aims to answer that question by summarizing and comparing the two models.

A.1. Sub-interface based configuration model overview

The key features of the sub-interface based configuration model can be summarized as:

* The model is primarily designed to enable layer 2 and layer 3 services on Ethernet interfaces that can be defined in a very flexible way to meet the varied requirements of service providers.

* Traffic is classified from an Ethernet-like interface to sub-interfaces based on fields in the layer 2 header. This is often based on VLAN Ids contained in the frame, but the model is extensible to other arbitrary fields in the frame header.

* Sub-interfaces are just a type of if:interface and hence support any feature configuration YANG models that can be applied generally to interfaces. For example, QoS or ACL models that reference if:interface can be applied to the sub-interfaces, or the sub-interface can be used as an Access Circuit in L2VPN or L3VPN models that reference if:interface.

* In the sub-interface based configuration model, the classification of traffic arriving on an interface to a given sub-interface, based on fields in the layer 2 header, is completely independent of how the traffic is forwarded. The sub-interface can be referenced (via references to if:interface) by other models that specify how traffic is forwarded; thus sub-interfaces can support multiple different forwarding paradigms, including but not limited to: layer 3 (IPv4/IPv6), layer 2 pseudowires (over MPLS or IP), VPLS instances, EVPN instance.

* The model is flexible in the scope of the VLAN Identifier space. I.e. by default VLAN Ids can be scoped locally to a single Ethernet-like trunk interface, but the scope is determined by the forwarding paradigm that is used.

A.2. IEEE 802.1Q Bridge Configuration Model Overview

The key features of the IEEE 802.1Q bridge configuration model can be summarized as:

* Each VLAN bridge component has a set of Ethernet interfaces that are members of that bridge. Sub-interfaces are not used, nor required in the 802.1Q bridge model.

* Within a VLAN bridge component, the VLAN tag in the packet is used, along with the destination MAC address, to determine how to forward the packet. Other forwarding paradigms are not supported by the 802.1Q model.

*Classification of traffic to a VLAN bridge component is based only on the Ethernet interface that it arrived on.

*VLAN Identifiers are scoped to a VLAN bridge component. Often devices only support a single bridge component and hence VLANs are scoped globally within the device.

*Feature configuration is specified in the context of the bridge, or particular VLANs on a bridge.

A.3. Possible Overlap Between the Two Models

Both models can be used for configuring similar basic layer 2 forwarding topologies. The 802.1Q bridge configuration model is optimised for configuring Virtual LANs that span across enterprises and data centers.

The sub-interface model can also be used for configuring equivalent Virtual LAN networks that span across enterprises and data centers, but often requires more configuration to be able to configure the equivalent constructs to the 802.1Q bridge model.

The sub-interface model really excels when implementing flexible L2 and L3 services, where those services may be handled on the same physical interface, and where the VLAN Identifier is being solely used to identify the customer or service that is being provided rather than a Virtual LAN. The sub-interface model provides more flexibility as to how traffic can be classified, how features can be applied to traffic streams, and how the traffic is to be forwarded.

Conversely, the 802.1Q bridge model can also be use to implement L2 services in some scenarios, but only if the forwarding paradigm being used to implement the service is the native Ethernet forwarding specified in 802.1Q - other forwarding paradigms such as pseudowires or VPLS are not supported. The 802.1Q bridge model does not implement L3 services at all, although this can be partly mitigated by using a virtual L3 interface construct that is a separate logical Ethernet-like interface which is a member of the bridge.

In conclusion, it is valid for both of these models to exist since they have different deployment scenarios for which they are optimized. Devices may choose which of the models (or both) to implement depending on what functionality the device is being designed for.

Authors' Addresses

Robert Wilton (editor)
Cisco Systems

Email: rwilton@cisco.com

Scott Mansfield (editor)
Ericsson

Email: scott.mansfield@ericsson.com