Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2015

R. Wilton, Ed.
D. Ball
G. Heron
Cisco Systems
March 9, 2015

Interface VLAN YANG Data Models
draft-wilton-intf-vlan-yang-00

Abstract

This document defines a YANG configuration data model for the management of VLAN sub-interfaces that augments the generic interfaces data model defined in RFC 7223 [RFC7223]. It provides support for basic tag matching to allow termination of an L2 VLAN segement into L3 services. It also provides support for flexible matching and rewriting of L2 header fields for L2 services.

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

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 http://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 September 10, 2015.

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG RFC 6020 [RFC6020] data model for the management of network interfaces. It defines interface-type specific extensions that augment the generic interfaces data model defined in RFC 7223 [RFC7223] to support configuration for VLAN sub-interfaces terminated to transport services at either layer 2 or layer 3.

It is defined as five separate YANG modules that each focus on a particular area of functionality. The YANG modules defined in this internet draft are:

dot1q-types.yang - Defines common types for identifying frames using fields from the 802.1Q VLAN tag

interface-common.yang - Defines common extensions to the IETF interface data model to support sub-interfaces

if-l3-vlan.yang - Defines the model for classifying L2 VLAN traffic to L3 transport services

flexible-encapsulation.yang - Defines the model for flexible classification of L2 traffic to L2 transport services

l2-bpdu-filtering.yang - Defines the model for implementing L2 BPDU filtering for VLAN services

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration (read-write), and "ro" means state data (read-only).
- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Objectives

The aim of of the YANG models contained in this draft is to provide the core model that is required to implement VLAN transport services on router based devices.

The secondary aim is to make the model cleanly extensible, both to handle greater depths of VLAN tag stacks if required, and also to allow vendors to extend the model to include additional forms of tag matching and rewriting if desired.

However, the intention is that it should not be necessary to have any vendor specific extentions to any of the YANG models defined in this document to implement standard Ethernet and VLAN services.

3. 802.1Q Types

The 802.1Q types YANG module contains type definitions for the basic fields defined in an 802.1Q VLAN tag. It also provides YANG groupings for identifying VLAN tags in various ways that can be used by other YANG modules where required.

4. Interfaces Common Model

The Interfaces Common model provides some basic extentions to the IETF interfaces YANG module for Ethernet and VLAN sub-interfaces.

The model provides:

- An encapsulation container and extensible choice statement for use by any interface types that allow for configurable L2 encapsulations.
- A configurable L2 MTU leaf applicable to all packet/frame based interfaces.
- A transport layer leaf to indicate whether the interface processes the traffic at L2 or L3.
- A parent interface leaf useable for all types of sub-interfaces that are bound to a particular parent interface.

The "interface-common" YANG module, has the following structure:

```
augment /if:interfaces/if:interface:
   +--rw encapsulation
      +--rw (encaps-type)?
augment /if:interfaces/if:interface:
   +--rw l2-mtu?   uint16
augment /if:interfaces/if:interface:
   +--rw transport-layer?   enumeration
augment /if:interfaces/if:interface:
   +--rw parent-interface?   if:interface-ref
```

5. L3 Interface VLAN Model

The L3 Interface VLAN model provides appropriate leaves for terminating an 802.1Q VLAN tagged segment to a sub-interface based L3 service. It allows for terminating of traffic with up to two 802.1Q VLAN tags.

The "if-l3-vlan" YANG module has the following structure:

```
augment /if:interfaces/if:interface/if-cmn:encapsulation/
  if-cmn:encaps-type:
   +--:(vlan)
      +--rw vlan
         +--rw tags
            +--rw tag* [index]
               +--rw index        uint8
               +--rw dot1q-tag
                  +--rw tag-type    dot1q-tag-type
                  +--rw vlan-id     dot1q-vlan-id
```

6. Flexible Encapsulation Model

The Flexible Encapsulation model is designed to allow for the flexible provisioning of layer 2 services. It provides the capability to classify 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 headers before the frame is handed off to the appropriate forwarding code for further handling.

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 traffic ingressing an interface then the reverse manipulation is always performed on traffic egressing 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 structure of the model is currently limited to matching or rewriting a maximum of two 802.1Q tags in the frame header but has

been designed to be easily extensible to matching/rewriting three or more VLAN tags in future, if required.

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.

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

```
augment /if:interfaces/if:interface/if-cmn:encapsulation/
  if-cmn:encaps-type:
   +--:(flexible) {flexible-encapsulation-rewrites}?
      +--rw flexible
         +--rw match
         |  +--rw (match-type)
         |     +--:(default)
         |     |  +--rw default?           empty
         |     +--:(untagged)
         |     |  +--rw untagged?          empty
         |     +--:(priority-tagged)
         |     |  +--rw priority-tagged
         |     |     +--rw tag-type?    dot1q:dot1q-tag-type
         |     +--:(vlan-tagged)
         |        +--rw vlan-tagged
         |           +--rw tag* [index]
         |           |  +--rw index        uint8
         |           |  +--rw dot1q-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 tag-rewrite {tag-rewrites}?
               |        +--rw pop-tags?    uint8
               |        +--rw push-tags* [index]
               |           +--rw index        uint8
               |           +--rw dot1q-tag
               |              +--rw tag-type    dot1q-tag-type
               |              +--rw vlan-id     dot1q-vlan-id
               +--:(asymmetrical) {asymmetric-rewrites}?
                  +--rw ingress
                  |  +--rw tag-rewrite {tag-rewrites}?
                  |     +--rw pop-tags?    uint8
```

```
                     |     +--rw push-tags* [index]
                     |        +--rw index        uint8
                     |        +--rw dot1q-tag
                     |           +--rw tag-type    dot1q-tag-type
                     |           +--rw vlan-id     dot1q-vlan-id
                     +--rw egress
                        +--rw tag-rewrite {tag-rewrites}?
                           +--rw pop-tags?    uint8
                           +--rw push-tags* [index]
                              +--rw index        uint8
                              +--rw dot1q-tag
                                 +--rw tag-type    dot1q-tag-type
                                 +--rw vlan-id     dot1q-vlan-id
   augment /if:interfaces/if:interface:
      +--rw flexible-encapsulation
         +--rw local-traffic-default-encaps
            +--rw tag* [index]
               +--rw index        uint8
               +--rw dot1q-tag
                  +--rw tag-type    dot1q-tag-type
                  +--rw vlan-id     dot1q-vlan-id
```

7. L2 BPDU Filtering

The L2 BPDU Filtering model adds a single configurable leaf to specify that BPDU filtering is in operation on a trunk interface.

The "l2-bpdu-filtering" YANG module has the following structure:

```
augment /if:interfaces/if:interface:
   +--rw bpdu
      +--rw filtering?   enumeration {bpdu-filtering}?
```

8. 802.1Q Types YANG Module

This YANG module has no external imports.

The expectation is that the raw 802.1Q VLAN tag fields types may end up being standardized in IEEE rather than IETF. They are included here to make the model complete.

However, the groupings that can be used to generally identify frames based on the fields in the 802.1Q tag would seem to fit with wherever the model resides.

```
<CODE BEGINS> file "dot1q-types@2015-02-26.yang"
module dot1q-types {
  namespace "urn:ietf:params:xml:ns:yang:dot1q-types";
```

```
     prefix dot1q;

     organization
       "Cisco Systems, Inc.
        Customer Service

        Postal: 170 W Tasman Drive
        San Jose, CA 95134

        Tel: +1 1800 553-NETS

        E-mail: cs-yang@cisco.com";

     contact
       "Robert Wilton - rwilton@cisco.com";

     description
       "This module contains a collection of generally useful YANG types
        that are specific to 802.1Q VLANs that can be usefully shared
        between multiple models.

        Terms and Acronyms

        802.1Q: IEEE 802.1Q VLANs

        VLAN (vlan): Virtual Local Area Network
        ";

     revision 2015-02-26 {
       description "Latest revision";

       reference "Internet-Draft draft-ietf-rwilton-vlan-yang-00.txt";
     }

     typedef PCP {
       type uint8 {
         range "0..7";
       }
       description
         "Priority Code Point. PCP is a 3-bit field that refers to the
          class of service applied to an 802.1Q VLAN tagged frame.  The
          field specifies a priority value between 0 and 7, these values
          can be used by quality of service (QoS) to prioritize
          different classes of traffic.";
       reference "IEEE 802.1Q (2014)";
     }

     /*
```

```
     * Defines what it means to be an 802.1Q VLAN Id, where values 0
     * and 4095 are reserved.
     */
    typedef dot1q-vlan-id {
      type uint16 {
        range "1..4094";
      }
      description "An 802.1Q VLAN Identifier";
      reference "IEEE 802.1Q (2014)";
    }

    /*
     * Defines the supported IEEE 802.1Q types that can be used for
     * VLAN tag matching.
     */
    identity dot1q-tag-vlan-type {
      description "Base identity from which all 802.1Q VLAN tag types
                   are derived from";
    }

    identity c-vlan {
      base dot1q-tag-vlan-type;
      description
        "An 802.1Q Customer-VLAN tag, normally using the 0x8100
         Ethertype";
    }

    identity s-vlan {
      base dot1q-tag-vlan-type;
      description
        "An 802.1Q Service-VLAN tag, using the 0x88a8 Ethertype
         originally introduced in 802.1ad, and incorporated into
         802.1Q (2011)";
    }

    typedef dot1q-tag-type {
      type identityref {
        base "dot1q-tag-vlan-type";
      }
      description "Identifies a specific 802.1Q tag type";
      reference "IEEE 802.1Q (2014)";
    }

    /*
     * Defines the type used to represent ranges of VLAN Ids.
     *
     * Ideally we would model that as a list of VLAN Ids in YANG, but
     * the model is easier to use if this is just represented as a
```

```
     * string.
     *
     * This type is used to match an ordered list of VLAN Ids, or
     * contiguous ranges of VLAN Ids.  Valid VLAN Ids must be in the
     * range 1 to 4094, and included in the list in non overlapping
     * ascending order.
     *
     * E.g. "1, 10-100, 50, 500-1000"
     */
    typedef dot1q-vlan-id-ranges {
      type string {
        pattern "([0-9]{1,4}(-[0-9]{1,4})?(,[0-9]{1,4}" +
                "(-[0-9]{1,4})?)*)";
      }
      description "A list of VLAN Ids, or non overlapping VLAN ranges,
                   in ascending order, between 1 and 4094";
    }

    /*
     * A grouping which represents an 802.1Q VLAN tag, matching both
     * the tag Ethertype and a single VLAN Id.  The PCP and DEI fields
     * in the 802.1Q tag are ignored for tag matching purposes.
     */
    grouping dot1q-tag {
      description "Grouping to allow configuration to identify a single
                   802.1Q VLAN tag";
      container dot1q-tag {
        description "Identifies an 802.1Q VLAN tag with an explicit
                     tag-type and a single VLAN Id";
        leaf tag-type {
          type dot1q-tag-type;
          mandatory true;
          description "VLAN tag type";
        }
        leaf vlan-id {
          type dot1q-vlan-id;
          mandatory true;
          description "VLAN Id";
        }
      }
    }

    /*
     * A grouping which represents an 802.1Q VLAN tag, matching both
     * the tag Ethertype and a single VLAN Id or "any" to match on any
     * VLAN Id.  The PCP and DEI fields in the 802.1Q tag are ignored
     * for tag matching purposes.
     */
```

```
    grouping dot1q-tag-or-any {
      description "Grouping to allow configuration to identify a single
                   802.1Q VLAN tag or the 'any' value to match any VLAN
                   Id not matched by a more specific VLAN Id match";
      container dot1q-tag {
        description "Identifies an 802.1Q VLAN tag with an explicit
                     tag-type and a single VLAN Id, or 'any' VLAN Id";
        leaf tag-type {
          type dot1q-tag-type;
          mandatory true;
          description "VLAN tag type";
        }
        leaf vlan-id {
          type union {
            type dot1q-vlan-id;
            type enumeration {
              enum "any" {
                value 4096;
                description
                  "Matches 'any' VLAN tag in the range 1 to 4094 that
                   is not matched by a more specific VLAN Id match";
              }
            }
          }
          mandatory true;
          description "VLAN Id or any";
        }
      }
    }

    /*
     * A grouping which represents an 802.1Q tag that matches a range
     * of VLAN Ids.  The PCP and DEI fields in the 802.1Q tag are
     * ignored for tag matching purposes.
     */
    grouping dot1q-tag-ranges {
      description "Grouping to allow configuration to identify an
                   802.1Q VLAN tag that matches any VLAN Id within a
                   set of non overlapping VLAN Id ranges";
      container dot1q-tag {
        description "Identifies an 802.1Q VLAN tag with an explicit
                     tag-type and and a range of VLAN Ids";
        leaf tag-type {
          type dot1q-tag-type;
          mandatory true;
          description "VLAN tag type";
        }
        leaf vlan-ids {
```

```
          type dot1q-vlan-id-ranges;
          mandatory true;
          description "VLAN Ids";
        }
      }
    }

    /*
     * A grouping which represents an 802.1Q VLAN tag, matching both
     * the tag Ethertype and a single VLAN Id, ordered list of ranges,
     * or "any" to match on any VLAN Id.  The PCP and DEI fields in the
     * 802.1Q tag are ignored for tag matching purposes.
     */
    grouping dot1q-tag-ranges-or-any {
      description "Grouping to allow configuration to identify an
                   802.1Q VLAN tag that matches any specific VLAN Id
                   within a set of non overlapping VLAN Id ranges, or
                   the 'any' value to match any VLAN Id";
      container dot1q-tag {
        description "Identifies an 802.1Q VLAN tag with an explicit
                     tag-type, an ordered list of VLAN Id ranges, or
                     'any' VLAN Id";
        leaf tag-type {
          type dot1q-tag-type;
          mandatory true;
          description "VLAN tag type";
        }
        leaf vlan-id {
          type union {
            type dot1q-vlan-id-ranges;
            type enumeration {
              enum "any" {
                description "Matches 'any' VLAN tag in the range 1 to
                             4094";
              }
            }
          }
          mandatory true;
          description "VLAN Ids or any";
        }
      }
    }
  }
  <CODE ENDS>
```

9. Interfaces Common YANG Module

This YANG module augments the interface container defined in RFC 7223 [RFC7223]

```
<CODE BEGINS> file "interfaces-common@2015-02-26.yang"
module interfaces-common {
  namespace "urn:ietf:params:xml:ns:yang:interfaces-common";
  prefix if-cmn;

  import ietf-interfaces {
    prefix if;
  }

  import iana-if-type {
    prefix ianaift;
  }

  organization
    "Cisco Systems, Inc.
     Customer Service

     Postal: 170 W Tasman Drive
     San Jose, CA 95134

     Tel: +1 1800 553-NETS

     E-mail: cs-yang@cisco.com";

  contact
    "Robert Wilton - rwilton@cisco.com";

  description
    "This module contains common definitions for extending the IETF
     interface YANG model (RFC 7223) with common configurable layer 2
     properties";

  revision 2015-02-26 {
    description "Latest revision";

    reference "Internet-Draft draft-ietf-rwilton-vlan-yang-00.txt";
  }

  /*
   * Various types of interfaces support a configurable layer 2
   * encapsulation, any that are supported by YANG should be
   * listed here.
   *
```

```
     * Different encapsulations can hook into the common encaps-type
     * choice statement.
     */
    augment "/if:interfaces/if:interface" {
      when "if:type = 'ianaift:ethernetCsmacd' or
            if:type = 'ianaift:ieee8023adLag' or
            if:type = 'ianaift:l2vlan'"  {
        description "All interface types that can have a configurable
                     L2 encapsulation";
      }
      description "Add encapsulation top level node to interface types
                   that support a configurable L2 encapsulation";

      container encapsulation {
        description
          "Holds the L2 encapsulation associated with an interfaces";
        choice encaps-type {
          description "Extensible choice of L2 encapsulations";
        }
      }
    }

    /*
     * Various types of interfaces support a configurable layer 2
     * MTU, all of them that are supported by YANG should be
     * listed here.
     */
    augment "/if:interfaces/if:interface" {
      when "if:type = 'ianaift:ethernetCsmacd' or
            if:type = 'ianaift:ieee8023adLag' or
            if:type = 'ianaift:l2vlan'"  {
        description "All interface types that can have a configurable
                     layer 2 MTU";
      }
      description "Add configurable layer-2 MTU to all appropriate
                   interface types";

      leaf l2-mtu {
        type uint16 {
          range "64 .. 65535";
        }
        description
          "The maximum size of layer 2 frame that may be transmitted
           or received on the interface (excluding any FCS overhead).
           In the case of Ethernet interfaces it also excludes the
           4-8 byte overhead of any known (i.e. explicitly matched by
           a child sub-interface) 801.1Q VLAN tags.";
      }
```

```
    }

    /*
     * Augments the IETF interfaces model with a leaf that indicates
     * whether traffic is to be transported as layer 2 or layer 3.
     *
     * All interface types that explicitly support forwarding frames
     * at layer 2 and that are supported by YANG should be listed here.
     *
     * Different encapsulation can hook into the common encaps-type
     * choice statement.
     */
    augment "/if:interfaces/if:interface" {
      when "if:type = 'ianaift:ethernetCsmacd' or
            if:type = 'ianaift:ieee8023adLag' or
            if:type = 'ianaift:l2vlan'" {
        description "Any interface types that support layer 2 transport
                     services";
      }
      description "Add a top level node to appropriate interfaces to
                   indicate which tranport layer an interface is
                   operating at";

      leaf transport-layer {
        type enumeration {
          enum layer-2 {
            value 2;
            description "Layer 2 transport";
          }
          enum layer-3 {
            value 3;
            description "Layer 3 transport";
          }
        }
        default layer-3;
        description
          "The transport layer at which the interface is operating at";
      }
    }

    /*
     * Add generic support for sub-interfaces.
     *
     * This should be extended to cover all interface types that are
     * child interfaces of other interfaces.
     */
    augment "/if:interfaces/if:interface" {
      when "if:type = 'ianaift:l2vlan'"  {
```

```
        description "Any sub-interfaces";
      }
      description "Add a parent interface field to interfaces to model
                   sub-interfaces";
      leaf parent-interface {
        type if:interface-ref;
        /*
         * TODO - How to make this mandatory without using the
         * mandatory keyword.
         * - Current options appear to be:
         *   - Possibly define a feature "parented-sub-interfaces".
         *   - Create a sub-interface container with presence.
         *   - Enforce the constraint with a must statement.
         */
        //mandatory true;
        description
          "This is the mandatory reference to the parent interface of
           this sub-interface.";
      }
    }
  }
  <CODE ENDS>
```

10. L3 Interface VLAN YANG Module

This YANG module augments the encapsultion container defined in the Interfaces Common YANG Module (Section 9)

```
<CODE BEGINS> file "if-l3-vlan@2015-02-26.yang"
module if-l3-vlan {
  namespace "urn:ietf:params:xml:ns:yang:if-l3-vlan";
  prefix if-l3-vlan;

  import ietf-interfaces {
    prefix if;
  }

  import iana-if-type {
    prefix ianaift;
  }

  import dot1q-types {
    prefix dot1q;
  }

  import interfaces-common {
    prefix if-cmn;
  }
```

```
     organization
       "Cisco Systems, Inc.
        Customer Service

        Postal: 170 W Tasman Drive
        San Jose, CA 95134

        Tel: +1 1800 553-NETS

        E-mail: cs-yang@cisco.com";

     contact
       "Robert Wilton - rwilton@cisco.com";

     description
       "This YANG module models L3 VLAN sub-interfaces
       ";

     revision 2015-02-26 {
       description "Latest revision";

       reference "Internet-Draft draft-ietf-rwilton-vlan-yang-00.txt";
     }

     feature l3-vlan-sub-interfaces {
       description
         "This feature indicates that the device supports L3 VLAN
          sub-interfaces";
     }

     /*
      * Add support for the 802.1Q VLAN encapsulation syntax on layer 3
      * terminated VLAN sub-interfaces.
      */
     augment "/if:interfaces/if:interface/if-cmn:encapsulation/" +
             "if-cmn:encaps-type" {
       when "../../if:type = 'ianaift:l2vlan' and
             ../../if-cmn:transport-layer = 'layer-3'" {
         description "Applies only to VLAN sub-interfaces that are
                      operating at layer 3";
       }
       if-feature l3-vlan-sub-interfaces;
       description "Augment the generic interface encapsulation with an
                    encapsulation for layer 3 VLAN sub-interfaces";

       /*
        * Matches a VLAN, or pair of VLAN Ids to classify traffic
        * into an L3 service.
```

```
       */
      case vlan {
        container vlan {
          description
          "Match VLAN tagged frames with specific VLAN Ids";
          container tags {
            description "Matches frames tagged with specific VLAN Ids";
            list tag {
              key "index";
              min-elements 1;
              max-elements 2;
              description "The tags to match, with the outermost tag to
                           match with index 0";
              leaf index {
                type uint8 {
                  range "0..1";
                }

                /*
                 * Only allow matching on an inner tag (at index 1), if
                 * also matching on the outer tag at the same time.
                 */
                must "index = 0 or
                      count(../../tag[index = 0]/index) > 0" {
                  error-message
                    "An inner tag can only be matched on when also
                     matching on an outer tag";
                  description
                    "Only allow matching on an inner tag, if also
                     matching on the outer tag at the same time";
                }
                description
                  "The index into the tag stack, outermost tag first";
              }

              uses dot1q:dot1q-tag;
            }
          }
        }
      }
    }
    <CODE ENDS>
```

11. Flexible Encapsulation YANG Module

This YANG module augments the encapsultion container defined in the Interfaces Common YANG Module (Section 9).

This YANG module also augments the interface container defined in RFC 7223 [RFC7223].

<CODE BEGINS> file "flexible-encapsulation@2015-02-26.yang"

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

  import ietf-interfaces {
    prefix if;
  }

  import iana-if-type {
    prefix ianaift;
  }

  import interfaces-common {
    prefix if-cmn;
  }

  import dot1q-types {
    prefix dot1q;
  }

  organization
    "Cisco Systems, Inc.
     Customer Service

     Postal: 170 W Tasman Drive
     San Jose, CA 95134

     Tel: +1 1800 553-NETS

     E-mail: cs-yang@cisco.com";

  contact
    "Robert Wilton - rwilton@cisco.com";

  description
    "This YANG module describes interface configuration for flexible
     VLAN matches and rewrites.";

  revision 2015-02-26 {
```

```
      description "Latest revision";

      reference "Internet-Draft draft-ietf-rwilton-vlan-yang-00.txt";
    }

    feature flexible-encapsulation-rewrites {
      description
        "This feature indicates whether the network element supports
         flexible Ethernet encapsulation that allows for matching VLAN
         ranges and performing independent tag manipulations";
    }

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

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

    feature tag-rewrites {
      description
        "This feature indicates whether the network element supports
         the flexible rewrite functionality specifying flexible tag
         rewrites";
    }

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

```
      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.";
          } // leaf default
        } // case default

        case untagged {
          description "Match untagged Ethernet frames only";
          leaf untagged {
            type empty;
            description
              "Untagged match.  Matches all untagged traffic.";
          } // leaf untagged
        } // case untagged

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

          container priority-tagged {
            description "Priority tag match";
            leaf tag-type {
              type dot1q:dot1q-tag-type;
              description "The 802.1Q tag type of matched priority
                           tagged packets";
            }
          }
        }

        case vlan-tagged {
          container vlan-tagged {
            description "Matches VLAN tagged frames";
            list tag {
              key "index";
              min-elements 1;
              max-elements 2;
              description "The tags to match, with the outermost tag to
                           match assigned index 0";
              leaf index {
```

```
                type uint8 {
                  range "0..1";
                }

                must "index = 0 or
                      count(../../tag[index = 0]/index) > 0" {
                  error-message "An inner tag can only be matched on
                                 when also matching on an outer tag";
                  description "Only allow matching on an inner tag, if
                               also matching on the outer tags at the
                               same time";
                }
                description
                  "The index into the tag stack, outermost tag first";
              }

              uses dot1q:dot1q-tag-ranges-or-any;
            }

            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.";
            }
          }
        }
      } // encaps-type
    }

    /*
     * Grouping for tag-rewrite that can be expressed either
     * symmetrically, or in the ingress and/or egress directions
     * independently.
     */
    grouping tag-rewrite {
      description "Flexible rewrite";
      leaf pop-tags {
        type uint8 {
          range 1..2;
        }
        description "The number of tags to pop (or translate if used in
                     conjunction with push-tags)";
      }
```

```
      list push-tags {
        key "index";
        max-elements 2;
        description "The number of tags to push (or translate if used
                     in conjunction with pop-tags)";
        /*
         * Server should order by increasing index.
         */
        leaf index {
          type uint8 {
            range 0..1;
          }

          /*
           * Only allow a push of an inner tag if an outer tag is also
           * being pushed.
           */
          must "index != 0 or
                count(../../push-tags[index = 0]/index) > 0" {
            error-message "An inner tag can only be pushed if an outer
                           tag is also specified";
            description "Only allow a push of an inner tag if an outer
                         tag is also being pushed";
          }
          description "The index into the tag stack";
        }

        uses dot1q:dot1q-tag;
      }
    }

    /*
     * Grouping for all flexible rewrites of fields in the L2 header.
     *
     * This currently only includes flexible tag rewrites, but is
     * designed to be extensible to cover rewrites of other fields in
     * the L2 header if required.
     */
    grouping flexible-rewrite {
      description "Flexible rewrite";

      /*
       * Tag rewrite.
       *
       * All tag rewrites are formed using a combination of pop-tags
       * and push-tags operations.
       */
      container tag-rewrite {
```

```
        if-feature tag-rewrites;
        description "Tag rewrite.  Translate operations are expressed
                     as a combination of tag push and pop operations.";
        uses tag-rewrite;
      }
    }

    augment "/if:interfaces/if:interface/if-cmn:encapsulation/" +
            "if-cmn:encaps-type" {
      when "../../if:type = 'ianaift:l2vlan' and
            ../../if-cmn:transport-layer = 'layer-2'" {
        description "Applies only to VLAN sub-interfaces that are
                     operating at transport layer 2";
      }
      description
        "Add flexible match and rewrite for VLAN sub-interfaces";

      /*
       * A flexible encapsulation allows for the matching of ranges and
       * sets of VLAN Ids.  The structure is also designed to be
       * extended to allow for matching/rewriting other fields within
       * the L2 frame header if required.
       */
      case flexible {
        if-feature flexible-encapsulation-rewrites;
        description "Flexible encapsulation and rewrite";
        container flexible {
          description "Flexible encapsulation and rewrite";

          container match {
            description
              "The match used to classify frames to this interface";
            uses flexible-match;
          }

          container rewrite {
            if-feature flexible-rewrites;
            description "L2 frame rewrite operations";
            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";
```

```
              }
            }

            /*
             * Allow asymmetrical rewrites to be specified.
             */
            case asymmetrical {
              if-feature asymmetric-rewrites;
              description "Asymmetrical rewrite";
              container ingress {
                uses flexible-rewrite;
                description "Ingress rewrite";
              }
              container egress {
                uses flexible-rewrite;
                description "Egress rewrite";
              }
            }
          }
        }
      }
    }

    augment "/if:interfaces/if:interface" {
      when "if:type = 'ianaift:l2vlan' and
            if-cmn:transport-layer = 'layer-2'" {
        description "Any L2 VLAN sub-interfaces";
      }
      description "Add flexible encapsulation configuration for VLAN
                   sub-interfaces";

      /*
       * All flexible encapsulation specific interface configuration
       * (except for the actual encapsulation and rewrite) is contained
       * by a flexible-encapsulation container on the interface.
       */
      container flexible-encapsulation {
        description
          "All per interface flexible encapsulation related fields";

        /*
         * For encapsulations that match a range of VLANs (or Any),
         * allow configuration to specify the default VLAN tag values
         * to use for any traffic that is locally sourced from an
         * interface on the device.
         */
        container local-traffic-default-encaps {
```

```
          description "The VLAN tags to use by default for locally
                       sourced traffic";
          list tag {
            key "index";
            max-elements 2;

            description
              "The VLAN tags to use by locally sourced traffic";

            leaf index {
              type uint8 {
                range "0..1";
              }

              /*
               * Only allow an inner tag to be specified if an outer
               * tag has also been specified.
               */
              must "index = 0 or
                    count(../../tag[index = 0]/index) > 0" {
                error-message "An inner tag can only be specified if an
                               outer tag has also been specified";
                description "Ensure that an inner tag cannot be
                             specified without an outer tag'";
              }

              description "The index into the tag stack, outermost tag
                           assigned index 0";
            }

            uses dot1q:dot1q-tag;
          }
        }
      }
    }
  }
  <CODE ENDS>
```

12. L2 BPDU filtering YANG Module

This YANG module augments the interface container defined in RFC 7223 [RFC7223] for Etherlike (Ethernet and 802.3 LAG (802.1AX) interfaces) trunk interfaces.

```
<CODE BEGINS> file "l2-bpdu-filtering@2015-02-26.yang"
module l2-bpdu-filtering {
  namespace "urn:ietf:params:xml:ns:yang:l2-bpdu-filtering";
  prefix bpdu;
```

```
     import ietf-interfaces {
       prefix if;
     }

     import iana-if-type {
       prefix ianaift;
     }

     organization
       "Cisco Systems, Inc.
        Customer Service

        Postal: 170 W Tasman Drive
        San Jose, CA 95134

        Tel: +1 1800 553-NETS

        E-mail: cs-yang@cisco.com";

     contact
       "Robert Wilton - rwilton@cisco.com";

     description
       "This YANG module describes the extentions for 802.1Q defined
        filtering of BPDUs via the destination MAC address.";

     revision 2015-02-26 {
       description "Latest revision";

       reference "Internet-Draft draft-ietf-rwilton-vlan-yang-00.txt";
     }

     feature bpdu-filtering {
       description
         "This feature indicates that the device supports standards
          compliant BPDU filtering";
     }

     /*
      * BPDU processing applies to all Etherlike interfaces.
      */
     augment "/if:interfaces/if:interface" {
       when "if:type = 'ianaift:ethernetCsmacd' or
             if:type = 'ianaift:ieee8023adLag'" {
         description "Applies to all Etherlike interfaces";
       }
       description "Add BPDU related configuration to Etherlike
                    interfaces";
```

```
      container bpdu {
        description "BPDU related configuration";

        /*
         * The filtering leaf defines the filtering of L2 BPDUs based
         * on their destination MAC address.  If no value has been
         * specified then the default behaviour is that there is no
         * filtering.
         */
        leaf filtering {
          if-feature bpdu-filtering;
          type enumeration {
            enum c-vlan {
              description "C-VLAN ingress frame filtering";
              reference
                "Table 8-1 C-VLAN and MAC Bridge component reserved
                 addresses of IEEE 802.1Q (2014)";
            }
            enum s-vlan {
              description "S-VLAN ingress frame filtering";
              reference
                "Table 8-2 S-VLAN component reserved addresses of
                 IEEE 802.1Q (2014)";
            }
            enum mac-relay {
              description "2-port MAC relay ingress frame filtering";
              reference
                "Table 8-3 TPMR component Reserved addresses of IEEE
                 802.1Q (2014)";
            }
          }
          description "The type of filtering to apply to all ingress
                       BPDU frames on this interface.  If no filtering
                       behavior is specified then frames are forwarded
                       by default unless they have been explicitly
                       peered by protocol specific configuration";
        }
      }
    }
  }
  <CODE ENDS>
```

13. Acknowledgements

The authors wish to thank Neil Ketley for his helpful comments contributing to this draft.

14. IANA Considerations

This document defines several new YANG module and the authors politely request that IANA assigns unique names to the YANG module files contained within this draft, and also appropriate URIs in the "IETF XML Registry".

15. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol RFC 6241 [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory to implement secure transport is SSH RFC 6242 [RFC6242]. The NETCONF access control model RFC 6536 [RFC6536] 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:

15.1. interfaces-common.yang

The interfaces-common YANG module contains a leaf to control the L2 MTU of an interface or sub-interface which if changed or deleted could cause traffic loss on the affected interface or sub-interfaces, or it could cause layer 2 tunnels to go down due to a mismatch in negotiated MTU. The following leaf is affected:

- interfaces/interface/l2-mtu

15.2. if-l3-vlan.yang

The nodes in the if-l3-vlan 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/vlan, that are sensitive to this are:

- tags

- tags/index
- tags/index/tag-type
- tags/index/vlan-id

15.3. 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
- priority-tagged
- priority-tagged/tag-type
- vlan-tagged
- vlan-tagged/index
- vlan-tagged/index/dot1q-tag/vlan-type
- vlan-tagged/index/dot1q-tag/vlan-id
- vlan-tagged/match-exact-tags

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/tag-rewrite/pop-tags
- symmetrical/tag-rewrite/push-tags

- symmetrical/tag-rewrite/push-tags/index
- symmetrical/tag-rewrite/push-tags/dot1q-tag/tag-type
- symmetrical/tag-rewrite/push-tags/dot1q-tag/vlan-id
- asymmetrical/ingress/tag-rewrite/pop-tags
- asymmetrical/ingress/tag-rewrite/push-tags
- asymmetrical/ingress/tag-rewrite/push-tags/index
- asymmetrical/ingress/tag-rewrite/push-tags/dot1q-tag/tag-type
- asymmetrical/ingress/tag-rewrite/push-tags/dot1q-tag/vlan-id
- asymmetrical/egress/tag-rewrite/pop-tags
- asymmetrical/egress/tag-rewrite/push-tags
- asymmetrical/egress/tag-rewrite/push-tags/index
- asymmetrical/egress/tag-rewrite/push-tags/dot1q-tag/tag-type
- asymmetrical/egress/tag-rewrite/push-tags/dot1q-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:

- tag
- tag/index
- tag/dot1q-tag/tag-type
- tag/dot1q-tag/vlan-id

15.4. l2-bpdu-filtering.yang

The l2-bpdu-filtering YANG module specifies a single leaf that defines what type of L2 BPDU filtering is in effect. Adding/modifying/deleting the following node could cause instabilities in L2 control protocols which could indirectly cause frame loss of network outages. Affected node:

- interfaces/interface/bpdu/filtering

16. References

16.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, May 2014.

[RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", RFC 7224, May 2014.

16.2. Informative References

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

Authors' Addresses

Robert Wilton (editor)
Cisco Systems

Email: rwilton@cisco.com

David Ball
Cisco Systems

Email: daviball@cisco.com

Giles Heron
Cisco Systems

Email: giheron@cisco.com