```
module ieee802-dot1q-types {
  namespace urn:ieee:std:802.1Q:yang:ieee802-dot1q-types;
  prefix dot1q-types;
  import ietf-yang-types {
    prefix yang;
  }
  organization
    "IEEE 802.1 Working Group";
  contact
    "WG-URL: http://www.ieee802.org/1/
    WG-EMail: stds-802-1-lL@ieee.org

    Contact: IEEE 802.1 Working Group Chair
    Postal: C/O IEEE 802.1 Working Group
            IEEE Standards Association
            445 Hoes Lane
            Piscataway, .O. Box 1331
            Piscataway
            NJ 08854
            USA

    E-mail: stdsSTDS-802-1-chairs@ieee.orgL@IEEE.ORG";
  description
    "Common types used within dot1Q-bridge modules.";
  revision 2020-06-04 {
    description
      "Published as part of IEEE Std 802.1Qcx-2020.
      Second version.";
    reference
      "IEEE Std 802.1Qcx-2020, Bridges and Bridged Networks -
      YANG Data Model for Connectivity Fault Management.";
  }
  revision 2018-03-07 {
    description
      "Published as part of IEEE Std 802.1Q-2018.
      Initial version.";
    reference
      "IEEE Std 802.1Q-2018, Bridges and Bridged Networks.";
  }

  identity dot1q-vlan-type {
    description
      "Base identity from which all 802.1Q VLAN tag types are derived
      from.";
  }
  identity c-vlan {
```

Explanation

1) This file illustrates item a) of the rogue comment on P802.1Qcw/D1.2 below by Johannes Specht.
2) Changes suggested to `ieee802-dot1q-types.yang` from P802.1Qcw/D1.2 are indicated in cyan and ~~magenta~~.
3) Changes between `ieee802-dot1q-types.yang` as published in IEEE Std 802.1Qcx-2020 and P802.1Qcw/D1.2 are indicated in ~~blue~~ and red.
4) File `ieee802-dot1q-types.yang` with the changes suggested by the Author is attached [i.e., not including the changes under item 3)].

Rogue Comment (suggested remedy per Comment)

a) Change the YANG code for nodes `traffic-class-table-grouping` and `traffic-class-table-grouping-v2` as indicated. In general, the commenter suggest to avoid identifier sufixes like "-v2", "-nextgen", "-fixed", or similar.
b) Re-base to `ieee802-dot1q-types.yang` from IEEE Std 802.1Qcx-2020.
c) Add a revision node for the expected IEEE Std 802.1Qcw-2021 above all other revision nodes.
d) Execute items b) and c) for all YANG source files attached to P802.1Qcw/D1.2.
e) Replace all references to `traffic-class-table-grouping-v2` by `traffic-class-table-grouping` in all YANG source files attached to P802.1Qcw/D1.2.
f) Update all UML diagrams and data scheme definitions as required by implementing item e).
g) Change ieee802-dot1q-vlan-pb to ieee802-dot1q-pb in line 20 page 183 of P802.1Qcw/D1.2.

```
  base dot1q-vlan-type;
  description
    "An 802.1Q Customer VLAN, using the 81-00 EtherType";
  reference
    "5.5 of IEEE Std 802.1Q-2018";
}
identity s-vlan {
  base dot1q-vlan-type;
  description
    "An 802.1Q Service VLAN, using the 88-A8 EtherType originally
    introduced in 802.1ad, and incorporated into 802.1Q (2011)";
  reference
    "5.6 of IEEE Std 802.1Q-2018";
}
typedef name-type {
  type string {
    length "0..32";
  }
  description
    "A text string of up to 32 characters, of locally determined
    significance.";
}
typedef port-number-type {
  type uint32 {
    range "1..65535";
  }
  description
    "The port number of the Bridge port for which this entry
    contains Bridge management information.";
}
typedef priority-type {
  type uint8 {
    range "0..7";
  }
  description
    "A range of priorities from 0 to 7 (inclusive). The Priority
    Code Point (PCP) is a 3-bit field that refers to the class of
    service associated with 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.";
}
typedef vid-range-type {
  type string {
    pattern
      "([1-9]"+
```

```
        "[0-9]{0,3}"+
        "(-[1-9][0-9]{0,3})?"+
        "(,[1-9][0-9]{0,3}(-[1-9][0-9]{0,3})?)*)";
    }
    description
      "A list of VLAN Ids, or non overlapping VLAN ranges, in
      ascending order, between 1 and 4094.

      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.

      For example: 1,10-100,250,500-1000";
  }
  typedef vlanid {
    type uint16 {
      range "1..4094";
    }
    description
      "The vlanid type uniquely identifies a VLAN. This is the 12-bit
      VLAN-ID used in the VLAN Tag header. The range is defined by the
      referenced specification. This type is in the value set and its
      semantics equivalent to the VlanId textual convention of the
      SMIv2.";
  }
  typedef vlan-index-type {
    type uint32 {
      range "1..4094 | 4096..4294967295";
    }
    description
      "A value used to index per-VLAN tables. Values of 0 and 4095 are
      not permitted. The range of valid VLAN indices. If the value is
      greater than 4095, then it represents a VLAN with scope local to
      the particular agent, i.e., one without a global VLAN-ID
      assigned to it. Such VLANs are outside the scope of IEEE 802.1Q,
      but it is convenient to be able to manage them in the same way
      using this YANG module.";
    reference
      "9.6 of IEEE Std 802.1Q-2018";
  }
  typedef mstid-type {
    type uint32 {
      range "1..4094";
    }
    description
```

```
    "In an MSTP Bridge, an MSTID, i.e., a value used to identify a
    spanning tree (or MST) instance";
  reference
    "13.8 of IEEE Std 802.1Q-2018";
}
typedef pcp-selection-type {
  type enumeration {
    enum 8P0D {
      description
        "8 priorities, 0 drop eligible";
    }
    enum 7P1D {
      description
        "7 priorities, 1 drop eligible";
    }
    enum 6P2D {
      description
        "6 priorities, 2 drop eligible";
    }
    enum 5P3D {
      description
        "5 priorities, 3 drop eligible";
    }
  }
  description
    "Priority Code Point selection types.";
  reference
    "12.6.2.5.3 of IEEE Std 802.1Q-2018
    6.9.3 of IEEE Std 802.1Q-2018";
}
typedef protocol-frame-format-type {
  type enumeration {
    enum Ethernet {
      description
        "Ethernet frame format";
    }
    enum rfc1042 {
      description
        "RFC 1042 frame format";
    }
    enum snap8021H {
      description
        "SNAP 802.1H frame format";
    }
    enum snapOther {
      description
```

```
        "Other SNAP frame format";
    }
    enum llcOther {
      description
        "Other LLC frame format";
    }
  }
  description
    "A value representing the frame format to be matched.";
  reference
    "12.10.1.7.1 of IEEE Std 802.1Q-2018";
}
typedef ethertype-type {
  type string {
    pattern "[0-9a-fA-F]{2}-[0-9a-fA-F]{2}";
  }
  description
    "The EtherType value represented in the canonical order defined
    by IEEE 802. The canonical representation uses uppercase
    characters.";
  reference
    "9.2 of IEEE Std 802-2014";
}
typedef dot1q-tag-type {
  type identityref {
    base dot1q-vlan-type;
  }
  description
    "Identifies a specific 802.1Q tag type";
  reference
    "IEEE Std 802.1Q-2018";
}
typedef traffic-class-type {
  type uint8 {
    range "0..7";
  }
  description
    "This is the numerical value associated with a traffic class in
    a Bridge. Larger values are associated with higher priority
    traffic classes.";
  reference
    "3.239 of IEEE Std 802.1Q-2018";
}
grouping dot1q-tag-classifier-grouping {
  description
    "A grouping which represents an 802.1Q VLAN, matching both the
```

```
    EtherType and a single VLAN Id.";
  leaf tag-type {
    type dot1q-tag-type;
    mandatory true;
    description
      "VLAN type";
  }
  leaf vlan-id {
    type vlanid;
    mandatory true;
    description
      "VLAN Id";
  }
}
grouping dot1q-tag-or-any-classifier-grouping {
  description
    "A grouping which represents an 802.1Q VLAN, matching both the
    EtherType and a single VLAN Id or 'any' to match on any VLAN Id.";
  leaf tag-type {
    type dot1q-tag-type;
    mandatory true;
    description
      "VLAN type";
  }
  leaf vlan-id {
    type union {
      type vlanid;
      type enumeration {
        enum any {
          value 4095;
          description
            "Matches 'any' VLAN in the range 1 to 4094 that is not
            matched by a more specific VLAN Id match";
        }
      }
    }
    mandatory true;
    description
      "VLAN Id or any";
  }
}
grouping dot1q-tag-ranges-classifier-grouping {
  description
    "A grouping which represents an 802.1Q VLAN that matches a range
    of VLAN Ids.";
  leaf tag-type {
```

```
    type dot1q-tag-type;
    mandatory true;
    description
      "VLAN type";
  }
  leaf vlan-ids {
    type vid-range-type;
    mandatory true;
    description
      "VLAN Ids";
  }
}
grouping dot1q-tag-ranges-or-any-classifier-grouping {
  description
    "A grouping which represents an 802.1Q VLAN, matching both the
    EtherType and a single VLAN Id, ordered list of ranges, or 'any'
    to match on any VLAN Id.";
  leaf tag-type {
    type dot1q-tag-type;
    mandatory true;
    description
      "VLAN type";
  }
  leaf vlan-id {
    type union {
      type vid-range-type;
      type enumeration {
        enum any {
          value 4095;
          description
            "Matches 'any' VLAN in the range 1 to 4094.";
        }
      }
    }
    mandatory true;
    description
      "VLAN Ids or any";
  }
}
grouping priority-regeneration-table-grouping {
  description
    "The priority regeneration table provides the ability to map
    incoming priority values on a per-Port basis, under management
    control.";
  reference
    "6.9.4 of IEEE Std 802.1Q-2018";
```

```
leaf priority0 {
  type priority-type;
  default "0";
  description
    "Priority 0";
  reference
    "12.6.2.3 of IEEE Std 802.1Q-2018
    6.9.4 of IEEE Std 802.1Q-2018";
}
leaf priority1 {
  type priority-type;
  default "1";
  description
    "Priority 1";
  reference
    "12.6.2.3 of IEEE Std 802.1Q-2018
    6.9.4 of IEEE Std 802.1Q-2018";
}
leaf priority2 {
  type priority-type;
  default "2";
  description
    "Priority 2";
  reference
    "12.6.2.3 of IEEE Std 802.1Q-2018
    6.9.4 of IEEE Std 802.1Q-2018";
}
leaf priority3 {
  type priority-type;
  default "3";
  description
    "Priority 3";
  reference
    "12.6.2.3 of IEEE Std 802.1Q-2018
    6.9.4 of IEEE Std 802.1Q-2018";
}
leaf priority4 {
  type priority-type;
  default "4";
  description
    "Priority 4";
  reference
    "12.6.2.3 of IEEE Std 802.1Q-2018
    6.9.4 of IEEE Std 802.1Q-2018";
}
leaf priority5 {
```

```
    type priority-type;
    default "5";
    description
      "Priority 5";
    reference
      "12.6.2.3 of IEEE Std 802.1Q-2018
      6.9.4 of IEEE Std 802.1Q-2018";
  }
  leaf priority6 {
    type priority-type;
    default "6";
    description
      "Priority 6";
    reference
      "12.6.2.3 of IEEE Std 802.1Q-2018
      6.9.4 of IEEE Std 802.1Q-2018";
  }
  leaf priority7 {
    type priority-type;
    default "7";
    description
      "Priority 7";
    reference
      "12.6.2.3 of IEEE Std 802.1Q-2018
      6.9.4 of IEEE Std 802.1Q-2018";
  }
}
grouping pcp-decoding-table-grouping {
  description
    "The Priority Code Point decoding table enables the decoding of
    the priority and drop-eligible parameters from the PCP.";
  reference
    "6.9.3 of IEEE Std 802.1Q-2018";
  list pcp-decoding-map {
    key "pcp";
    description
      "This map associates the priority code point field found in
      the VLAN to a priority and drop eligible value based upon the
      priority code point selection type.";
    leaf pcp {
      type pcp-selection-type;
      description
        "The priority code point selection type.";
      reference
        "12.6.2.7 of IEEE Std 802.1Q-2018
        6.9.3 of IEEE Std 802.1Q-2018";
```

```
    }
    list priority-map {
      key "priority-code-point";
      description
        "This map associated a priority code point value to priority
        and drop eligible parameters.";
      leaf priority-code-point {
        type priority-type;
        description
          "Priority associated with the pcp.";
        reference
          "12.6.2.7 of IEEE Std 802.1Q-2018
          6.9.3 of IEEE Std 802.1Q-2018";
      }
      leaf priority {
        type priority-type;
        description
          "Priority associated with the pcp.";
        reference
          "12.6.2.7 of IEEE Std 802.1Q-2018
          6.9.3 of IEEE Std 802.1Q-2018";
      }
      leaf drop-eligible {
        type boolean;
        description
          "Drop eligible value for pcp";
        reference
          "12.6.2.7 of IEEE Std 802.1Q-2018
          6.9.3 of IEEE Std 802.1Q-2018";
      }
    }
  }
}
grouping pcp-encoding-table-grouping {
  description
    "The Priority Code Point encoding table encodes the priority and
    drop-eligible parameters in the PCP field of the VLAN tag.";
  reference
    "12.6.2.9 of IEEE Std 802.1Q-2018
    6.9.3 of IEEE Std 802.1Q-2018";
  list pcp-encoding-map {
    key "pcp";
    description
      "This map associated the priority and drop-eligible parameters
      to the priority used to encode the PCP of the VLAN based upon
      the priority code point selection type.";
```

```
    leaf pcp {
      type pcp-selection-type;
      description
        "The priority code point selection type.";
      reference
        "12.6.2.7 of IEEE Std 802.1Q-2018
        6.9.3 of IEEE Std 802.1Q-2018";
    }
    list priority-map {
      key "priority dei";
      description
        "This map associated the priority and drop-eligible
        parameters to the priority code point field of the VLAN tag.";
      leaf priority {
        type priority-type;
        description
          "Priority associated with the pcp.";
        reference
          "12.6.2.7 of IEEE Std 802.1Q-2018
          6.9.3 of IEEE Std 802.1Q-2018";
      }
      leaf dei {
        type boolean;
        description
          "The drop eligible value.";
        reference
          "12.6.2 of IEEE Std 802.1Q-2018
          8.6.6 of IEEE Std 802.1Q-2018";
      }
      leaf priority-code-point {
        type priority-type;
        description
          "PCP value for priority when DEI value";
        reference
          "12.6.2.9 of IEEE Std 802.1Q-2018
          6.9.3 of IEEE Std 802.1Q-2018";
      }
    }
  }
}
grouping service-access-priority-table-grouping {
  description
    "The Service Access Priority Table associates a received
    priority with a serice access priority.";
  reference
    "12.6.2.17 of IEEE Std 802.1Q-2018
```

```
    6.13.1 of IEEE Std 802.1Q-2018";
  leaf priority0 {
    type priority-type;
    default "0";
    description
      "Service access priority value for priority 0";
    reference
      "12.6.2.17 of IEEE Std 802.1Q-2018
      6.13.1 of IEEE Std 802.1Q-2018";
  }
  leaf priority1 {
    type priority-type;
    default "1";
    description
      "Service access priority value for priority 1";
    reference
      "12.6.2.17 of IEEE Std 802.1Q-2018
      6.13.1 of IEEE Std 802.1Q-2018";
  }
  leaf priority2 {
    type priority-type;
    default "2";
    description
      "Service access priority value for priority 2";
    reference
      "12.6.2.17 of IEEE Std 802.1Q-2018
      6.13.1 of IEEE Std 802.1Q-2018";
  }
  leaf priority3 {
    type priority-type;
    default "3";
    description
      "Service access priority value for priority 3";
    reference
      "12.6.2.17 of IEEE Std 802.1Q-2018
      6.13.1 of IEEE Std 802.1Q-2018";
  }
  leaf priority4 {
    type priority-type;
    default "4";
    description
      "Service access priority value for priority 4";
    reference
      "12.6.2.17 of IEEE Std 802.1Q-2018
      6.13.1 of IEEE Std 802.1Q-2018";
  }
```

```
    leaf priority5 {
      type priority-type;
      default "5";
      description
        "Service access priority value for priority 5";
      reference
        "12.6.2.17 of IEEE Std 802.1Q-2018
        6.13.1 of IEEE Std 802.1Q-2018";
    }
    leaf priority6 {
      type priority-type;
      default "6";
      description
        "Service access priority value for priority 6";
      reference
        "12.6.2.17 of IEEE Std 802.1Q-2018
        6.13.1 of IEEE Std 802.1Q-2018";
    }
    leaf priority7 {
      type priority-type;
      default "7";
      description
        "Service access priority value for priority 7";
      reference
        "12.6.2.17 of IEEE Std 802.1Q-2018
        6.13.1 of IEEE Std 802.1Q-2018";
    }
  }
  grouping traffic-class-table-grouping {
    status obsolete;
    description
      "The Traffic Class Table models the operations that can be
      performed on, or inquire about, the current contents of the
      Traffic Class Table (8.6.6) for a given Port. This grouping
      modeled the Traffic Class Table incorrectly, and therefore its
      status is obsolete. It is replaced by
      traffic-class-table-grouping-v2";
    reference
      "12.6.3 of IEEE Std 802.1Q-2018
      8.6.6 of IEEE Std 802.1Q-2018";
    list traffic-class-map {
      key "priority";
      status deprecated;
      description
        "The priority index into the traffic class table. This list
        modeled the Traffic Class Table incorrectly, and therefore
```

```
        its status is obsolete. It is replaced by sibling container
        traffic-class-table.";
      leaf priority {
        type priority-type;
        description
          "The priority of the traffic class entry.";
        reference
          "8.6.6 of IEEE Std 802.1Q-2018";
      }
      list available-traffic-class {
        key "num-traffic-class";
        description
          "The traffic class index associated with a given priority
          within the traffic class table.";
        reference
          "8.6.6 of IEEE Std 802.1Q-2018";
        leaf num-traffic-class {
          type uint8 {
            range "1..8";
          }
          description
            "The available number of traffic classes.";
          reference
            "8.6.6 of IEEE Std 802.1Q-2018";
        }
        leaf traffic-class {
          type traffic-class-type;
          description
            "The traffic class index associated with a given traffic
            class entry.";
          reference
            "8.6.6 of IEEE Std 802.1Q-2018";
        }
      }
    }
  }
  grouping traffic-class-table-grouping-v2 {
    description
      "The default values for the table are specified in 8.6.6.";
    reference
      "12.6.2 of IEEE Std 802.1Q-2018";
    container traffic-class-table {
      description
        "The priority to traffic class mapping.";
      reference
        "8.6.6 of IEEE Std 802.1Q-2018";
```

```
      leaf number-of-traffic-classes {
        type uint8 {
          range "1..8";
        }
        description
          "The number of egress traffic classes supported on this
          port. This object may be optionally be read-only.";
        reference
          "12.6.3.1 of IEEE Std 802.1Q-2018";
      }
      leaf priority0 {
        type traffic-class-type;
        must "current() < ../number-of-traffic-classes";
        description
          "The traffic class index associated with priority 0";
        reference
          "8.6.6 of IEEE Std 802.1Q-2018";
      }
      leaf priority1 {
        type traffic-class-type;
        must "current() < ../number-of-traffic-classes";
        description
          "The traffic class index associated with priority 1";
        reference
          "8.6.6 of IEEE Std 802.1Q-2018";
      }
      leaf priority2 {
        type traffic-class-type;
        must "current() < ../number-of-traffic-classes";
        description
          "The traffic class index associated with priority 2";
        reference
          "8.6.6 of IEEE Std 802.1Q-2018";
      }
      leaf priority3 {
        type traffic-class-type;
        must "current() < ../number-of-traffic-classes";
        description
          "The traffic class index associated with priority 3";
        reference
          "8.6.6 of IEEE Std 802.1Q-2018";
      }
      leaf priority4 {
        type traffic-class-type;
        must "current() < ../number-of-traffic-classes";
        description
```

```
          "The traffic class index associated with priority 4";
        reference
          "8.6.6 of IEEE Std 802.1Q-2018";
      }
      leaf priority5 {
        type traffic-class-type;
        must "current() < ../number-of-traffic-classes";
        description
          "The traffic class index associated with priority 5";
        reference
          "8.6.6 of IEEE Std 802.1Q-2018";
      }
      leaf priority6 {
        type traffic-class-type;
        must "current() < ../number-of-traffic-classes";
        description
          "The traffic class index associated with priority 6";
        reference
          "8.6.6 of IEEE Std 802.1Q-2018";
      }
      leaf priority7 {
        type traffic-class-type;
        must "current() < ../number-of-traffic-classes";
        description
          "The traffic class index associated with priority 7";
        reference
          "8.6.6 of IEEE Std 802.1Q-2018";
      }
    }
  }
  grouping port-map-grouping {
    description
      "A set of control indicators, one for each Port. A Port Map,
      containing a control element for each outbound Port";
    reference
      "8.8.1 of IEEE Std 802.1Q-2018
      8.8.2 of IEEE Std 802.1Q-2018";
    list port-map {
      key "port-ref";
      description
        "The list of entries composing the port map.";
      leaf port-ref {
        type port-number-type;
        description
          "The interface port reference associated with this map.";
        reference
```

```
    "8.8.1 of IEEE Std 802.1Q-2018";
}
choice map-type {
  description
    "Type of port map";
  container static-filtering-entries {
    description
      "Static filtering entries attributes.";
    leaf control-element {
      type enumeration {
        enum forward {
          description
            "Forwarded, independently of any dynamic filtering
            information held by the FDB.";
        }
        enum filter {
          description
            "Filtered, independently of any dynamic filtering
            information.";
        }
        enum forward-filter {
          description
            "Forwarded or filtered on the basis of dynamic
            filtering information, or on the basis of the
            default Group filtering behavior for the outbound
            Port (8.8.6) if no dynamic filtering information is
            present specifically for the MAC address.";
        }
      }
      description
        "containing a control element for each outbound Port,
        specifying that a frame with a destination MAC address,
        and in the case of VLAN Bridge components, VID that
        meets this specification.";
      reference
        "8.8.1 of IEEE Std 802.1Q-2018";
    }
    leaf connection-identifier {
      type port-number-type;
      description
        "A Port MAP may contain a connection identifier (8.8.12)
        for each outbound port. The connection identifier may be
        associated with the Bridge Port value maintained in a
        Dynamic Filtering Entry of the FDB for Bridge Ports.";
      reference
        "8.8.1 of IEEE Std 802.1Q-2018
```

```
      8.8.12 of IEEE Std 802.1Q-2018";
  }
}
container static-vlan-registration-entries {
  description
    "Static VLAN registration entries.";
  leaf registrar-admin-control {
    type enumeration {
      enum fixed-new-ignored {
        description
          "Registration Fixed (New ignored).";
      }
      enum fixed-new-propagated {
        description
          "Registration Fixed (New propagated.";
      }
      enum forbidden {
        description
          "Registration Forbidden.";
      }
      enum normal {
        description
          "Normal Registration.";
      }
    }
    description
      "The Registrar Administrative Control values for MVRP
      and MIRP for the VID.";
    reference
      "8.8.2 of IEEE Std 802.1Q-2018";
  }
  leaf vlan-transmitted {
    type enumeration {
      enum tagged {
        description
          "VLAN-tagged";
      }
      enum untagged {
        description
          "VLAN-untagged";
      }
    }
    description
      "Whether frames are to be VLAN-tagged or untagged when
      transmitted.";
    reference
```

```
      "8.8.2 of IEEE Std 802.1Q-2018";
  }
}
container mac-address-registration-entries {
  description
    "MAC address registration entries attributes.";
  leaf control-element {
    type enumeration {
      enum registered {
        description
          "Forwarded, independently of any dynamic filtering
          information held by the FDB.";
      }
      enum not-registered {
        description
          "Filtered, independently of any dynamic filtering
          information.";
      }
    }
    description
      "containing a control element for each outbound Port,
      specifying that a frame with a destination MAC address,
      and in the case of VLAN Bridge components, VID that
      meets this specification.";
    reference
      "8.8.4 of IEEE Std 802.1Q-2018";
  }
}
container dynamic-vlan-registration-entries {
  description
    "Dynamic VLAN registration entries attributes.";
  leaf control-element {
    type enumeration {
      enum registered {
        description
          "Forwarded, independently of any dynamic filtering
          information held by the FDB.";
      }
    }
    description
      "containing a control element for each outbound Port,
      specifying that a frame with a destination MAC address,
      and in the case of VLAN Bridge components, VID that
      meets this specification.";
    reference
      "8.8.5 of IEEE Std 802.1Q-2018";
```

```
  }
}
container dynamic-reservation-entries {
  description
    "Dynamic reservation entries attributes.";
  leaf control-element {
    type enumeration {
      enum forward {
        description
          "Forwarded, independently of any dynamic filtering
          information held by the FDB.";
      }
      enum filter {
        description
          "Filtered, independently of any dynamic filtering
          information.";
      }
    }
    description
      "Containing a control element for each outbound Port,
      specifying that a frame with a destination MAC address,
      and in the case of VLAN Bridge components, VID that
      meets this specification.";
    reference
      "8.8.7 of IEEE Std 802.1Q-2018";
  }
}
container dynamic-filtering-entries {
  description
    "Dynamic filtering entries attributes.";
  leaf control-element {
    type enumeration {
      enum forward {
        description
          "Forwarded, independently of any dynamic filtering
          information held by the FDB.";
      }
    }
    description
      "Containing a control element for each outbound Port,
      specifying that a frame with a destination MAC address,
      and in the case of VLAN Bridge components, VID that
      meets this specification.";
    reference
      "8.8.3 of IEEE Std 802.1Q-2018";
  }
```

```
      }
    }
  }
}
grouping bridge-port-statistics-grouping {
  description
    "Grouping of bridge port statistics.";
  reference
    "12.6.1.1.3 of IEEE Std 802.1Q-2018";
  leaf delay-exceeded-discards {
    type yang:counter64;
    description
      "The number of frames discarded by this port due to excessive
      transit delay through the Bridge. It is incremented by both
      transparent and source route Bridges.";
    reference
      "12.6.1.1.3 of IEEE Std 802.1Q-2018
      8.6.6 of IEEE Std 802.1Q-2018";
  }
  leaf mtu-exceeded-discards {
    type yang:counter64;
    description
      "The number of frames discarded by this port due to an
      excessive size. It is incremented by both transparent and
      source route Bridges.";
    reference
      "12.6.1.1.3, item g) of IEEE Std 802.1Q-2018";
  }
  leaf frame-rx {
    type yang:counter64;
    description
      "The number of frames that have been received by this port
      from its segment. Note that a frame received on the interface
      corresponding to this port is only counted by this object if
      and only if it is for a protocol being processed by the local
      bridging function, including Bridge management frames.";
    reference
      "12.6.1.1.3 of IEEE Std 802.1Q-2018";
  }
  leaf octets-rx {
    type yang:counter64;
    description
      "The total number of octets in all valid frames received
      (including BPDUs, frames addressed to the Bridge as an end
      station, and frames that were submitted to the Forwarding
      Process).";
```

```
  reference
    "12.6.1.1.3 of IEEE Std 802.1Q-2018";
}
leaf frame-tx {
  type yang:counter64;
  description
    "The number of frames that have been transmitted by this port
    to its segment. Note that a frame transmitted on the interface
    corresponding to this port is only counted by this object if
    and only if it is for a protocol being processed by the local
    bridging function, including Bridge management frames.";
}
leaf octets-tx {
  type yang:counter64;
  description
    "The total number of octets that have been transmitted by this
    port to its segment.";
}
leaf discard-inbound {
  type yang:counter64;
  description
    "Count of received valid frames that were discarded (i.e.,
    filtered) by the Forwarding Process.";
  reference
    "12.6.1.1.3 of IEEE Std 802.1Q-2018";
}
leaf forward-outbound {
  type yang:counter64;
  description
    "The number of frames forwarded to the associated MAC Entity
    (8.5).";
  reference
    "12.6.1.1.3 of IEEE Std 802.1Q-2018";
}
leaf discard-lack-of-buffers {
  type yang:counter64;
  description
    "The count of frames that were to be transmitted through the
    associated Port but were discarded due to lack of buffers.";
  reference
    "12.6.1.1.3 of IEEE Std 802.1Q-2018";
}
leaf discard-transit-delay-exceeded {
  type yang:counter64;
  description
    "The number of frames discarded by this port due to excessive
```

```
        transit delay through the Bridge. It is incremented by both
        transparent and source route Bridges.";
      reference
        "12.6.1.1.3 of IEEE Std 802.1Q-2018";
    }
    leaf discard-on-error {
      type yang:counter64;
      description
        "The number of frames that were to be forwarded on the
        associated MAC but could not be transmitted (e.g., frame would
        be too large, 6.5.8).";
      reference
        "12.6.1.1.3 of IEEE Std 802.1Q-2018";
    }
  }
}

identity type-of-operation {
  description
    "Represents the operation type (name).";
}
grouping base-gate-control-entries {
  list gate-control-entry {
    key "index";
    leaf index {
      type uint32;
    }
    leaf operation-name {
      type identityref {
        base type-of-operation;
      }
      mandatory true;
      description
        "The name (type) of the operation for this entry.";
    }
    leaf time-interval-value {
      type uint32;
      description
        "timeIntervalValue is a 32-bit unsigned integer,
        representing a number of nanoseconds. After timeIntervalValue
        nanoseconds have elapsed since the completion of the previous
        entry in the gate control list, control passes to the next
        entry.";
      reference
        "12.29.1.2.3 of IEEE Std 802.1Q-2018 8.6.8.4 of IEEE Std
```

```
            802.1Q-2018";
        }
    }
  }
}
```