P802.1Qdy - RSTP/MSTP YANG – Reusability across SDOs
Murugan Balraj (Nokia)
(murugan.balraj@nokia.com)
Introduction

This presentation

• Highlights a problem in the proposed YANG model for RSTP, MSTP as part of P802.1Qdy that would prevent its reuse in other SDOs like BBF.

• Presents a possible solution that would allow the YANG model to be reused by other SDOs, but at the same time preserve the overall structure and functionality of the YANG from IEEE 802.1 perspective.

• Highlights some inconsistencies in the data that could arise and possible solution.
The problem (RSTP)

48.6 YANG modules

Insert 48.6.26 after 48.6.25 (inserted by IEEE Std 802.1Qdx-2024) as follows:

48.6.26 The iee802-dot1q-rstp YANG module

define the iee802-dot1q-rstp { 
    yang-version 1.1;
    namespace "urn:ieee:std:802.1Q:yang:ieee802-dot1q-rstp";
    prefix rstp;

    augment "/dot1q:bridges/dot1q:bridge/dot1q:component"
        description "Augment Bridge with RSTP configuration."
        reference "13.24, 13.25, and 13.26 of IEEE Std 802.1Q."
        container rstp {
            presence "The presence of this container indicates that RSTP is supported";
        }

    augment "/if:interfaces/if:interface/dot1q:bridge-port"
        description "Augment Bridge Port with RSTP configuration"
        reference "13.24, 13.25, and 13.27 of IEEE Std 802.1Q."
        container rstp {
            presence "The presence of this container indicates that RSTP is supported";
        }

The “augments” clause present in the YANG module cannot be overridden and this makes it impossible for other SDOs to use this YANG module since they may not have the notion of Bridge / Component.
The problem (MSTP)

The “augments” clause present in the YANG module cannot be overridden and this makes it impossible for other SDOs to use this YANG module since they may not have the notion of Bridge / Component.
IEEE 802.1 Provider Edge Bridge

Figure 15-6—Customer Edge Ports (CEPs)

BBF Layer-2 Access device

S-VLAN

network

sub-interface 1
“network 200”
VLAN-ID 200

forwarder-port
“network 200”

forwarder-port
“user 100 to
network 200”

forwarding
database

C-VLAN

user

sub-interface 2
“user 100” VLAN-
ID 100

PTM 1

G.fast 1

In a BBF Access device, the SVLAN component and CVLAN components are not logically modelled separately but they have forwarder ports and all the SVLAN and CVLAN aware forwarder ports could be part of the same “forwarder”. 

Reference:
IEEE 802.1Q-2022, Section “15.4 C-tagged service interface”, Figure 15-6 – Customer Edge Ports (CEP)
BBF Layer-2 Access device YANG view (hardware/component/interface/L2-forwarding)

BBF specific YANG modules augments and/or leafref the IETF hardware/component, IETF system and IETF interfaces YANG modules. /hw:hardware/hw:component /sys:system /if:interfaces-state/if:interface

Reference: Common YANG Modules for Access Networks (broadband-forum.org), Page 13, section 1.2, Figure 1 – YANG Data Model Relationships

Proven strategy for reusability across SDOs

The YANG module that other SDOs like BBF would use in their devices has to be free of “augment” clauses and references to Bridge / Component / Bridge Port.

- Split the YANG module into two modules. The Base YANG module would contain the protocol specific YANG definitions and an IEEE 802.1 specific module would augment this base module and contain the “augment” clauses.

- Other SDOs like BBF could then use just the Base YANG module.

Following are some examples of existing IEEE 802.1 YANG models that are reusable

(Path: https://github.com/YangModels/yang/blob/main/standard/ieee/published/802.1)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Common reusable YANG</th>
<th>IEEE 802.1 Bridge specific YANG</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFM</td>
<td>ieee802-dot1q-cfm.yang</td>
<td>ieee802-dot1q-cfm-bridge.yang</td>
</tr>
<tr>
<td>Scheduler</td>
<td>ieee802-dot1q-sched.yang</td>
<td>ieee802-dot1q-sched-bridge.yang</td>
</tr>
<tr>
<td>PSFP</td>
<td>ieee802-dot1q-psfp.yang</td>
<td>ieee802-dot1q-psfp-bridge.yang</td>
</tr>
<tr>
<td>Stream Filters and Gates</td>
<td>ieee802-dot1q-stream-filters-gates.yang</td>
<td>ieee802-dot1q-stream-filters-gates-bridge.yang</td>
</tr>
<tr>
<td>Preemption</td>
<td>ieee802-dot1q-preemption.yang</td>
<td>ieee802-dot1q-preemption-bridge.yang</td>
</tr>
<tr>
<td>Congestion-isolation</td>
<td>ieee802-dot1q-congestion-isolation.yang</td>
<td>ieee802-dot1q-congestion-isolation-bridge.yang</td>
</tr>
</tbody>
</table>
Example of some existing IEEE 802.1 YANG models

### IEEE 802.1 Base CFM model

*(Protocol specific YANG definitions)*

| augment | ieee802-dot1q-cfm.yang |

### IEEE 802.1 Bridge specific CFM augmentation

augment “/dot1q-cfm:cfm/dot1q-cfm:mac-maintenance-group”
---bridge-name -> leafref to IEEE 802.1 Bridge
---bridge-component-name -> leafref to IEEE 802.1 Bridge component

augment “/dot1q-cfm:cfm/dot1q-cfm:mac-maintenance-group/dot1q-cfm:mep”
---interface-ref -> leafref to IETF interface

| augment | ieee802-dot1q-cfm-bridge.yang |

### BBF specific CFM augmentation

augment “/dot1q-cfm:cfm/dot1q-cfm:mac-maintenance-group”
---forwarder -> leafref to BBF Forwarder

augment “/dot1q-cfm:cfm/dot1q-cfm:mac-maintenance-group/dot1q-cfm:mep”
---interface-ref -> leafref to BBF vlan-sub-interface

| augment | bbf-dot1q-cfm-interfaces.yang | bbf-dot1q-cfm-l2-forwarding.yang |

---

### IEEE 802.1 Base CFM model + IEEE 802.1 Bridge specific CFM augmentation

Together builds the complete CFM model for the devices conforming to the IEEE 802.1 specifications.

### IEEE 802.1 Base CFM model + BBF specific CFM augmentation

Together builds the complete CFM model for the devices conforming to the BBF specifications.

### IEEE 802.1 Base CFM model + ITU-T Y.1731 specific CFM augmentation

Together builds the complete model for devices conforming to the ITU-T specifications.

Ref: Recommendation ITU-T G.8052

---

Note: Recommendation ITU-T Y.1731 CFM also uses the IEEE 802.1 Base CFM model to augment ITU-T Y.1731 specific CFM YANG definitions.
Example of some existing IEEE 802.1 YANG Models

The IEEE 802.1 Bridge specific YANG module has a “uses” statement that augments the “Component” with the “grouping” from the base module.

**IEEE 802.1 Bridge specific augmentation**

```yaml
iee802-dot1q-psfp-bridge.yang

augment "/dot1q:bridges/dot1q:bridge/dot1q:component {
  uses psfp:psfp-parameters
}
```

The base YANG module uses the “grouping” statement.

Other SDO specific augmentation

Other SDOs like BBF could also utilize the “uses” statement to include the “grouping”.

Other YANG models “**Scheduler, Stream Filters and Gates, Preemption and Congestion-isolation**” also follow the same approach.

The Base module has a “**grouping**” statement and the SDO specific module achieves the augmentation with the “**uses**” statement.
P802.1Qdy decoupling proposal – overall strategy

Implementation in a device conforming to IEEE 802.1 specifications

Device conforming to IEEE 802.1 specifications would implement the YANG model of:

1. Protocol specific aspects YANG definition (reusable across SDOs)
2. IEEE 802.1 device specific YANG augmentation
P802.1Qdy decoupling proposal – overall strategy

Implementation in other SDOs

Device conforming to BBF specifications would implement the YANG model of: 1 & 3
IEEE 802.1Qdy decoupling proposal – more details

IEEE 802.1Qdy current YANG definition

IEEE 802.1 device specific YANG module which uses Base YANG module

Can be reused in other SDOs like BBF

Base YANG module – MSTP protocol specific YANG definitions as YANG grouping

IEEE 802.1 device specific augments YANG definition

Per component MSTP augment

YANG module "ieee802-dot1q-mstp-bridge.yang":
- augment "/dot1q:bridges/dot1q:bridge/dot1q:component/dot1q:bridge-mst":
  - uses mstp-per-component-protocol-specific-data

Per interface MSTP augment

YANG module "ieee802-dot1q-mstp-bridge.yang":
- augment "/if:interfaces/if:interface/dot1q:bridge-port":
  - uses mstp-per-interface-protocol-specific-data

YANG module "ieee802-dot1q-mstp.yang":
- YANG “Grouping” (mstp-per-component-protocol-specific-data):
  - MSTP YANG configuration/state data which are per component specific.

YANG module "ieee802-dot1q-mstp.yang":
- YANG “Grouping” (mstp-per-interface-protocol-specific-data):
  - MSTP YANG configuration/state data which are per interface specific.
P802.1Qdy decoupling proposal – MSTP YANG

Note: Only the **YANG** is restructured considering reusability. The resultant **YANG** tree/functionality is the same as that on page 20 of 802-1Qdy-d2-0.pdf
P802.1Qdy decoupling proposal – MSTP YANG

Resultant YANG tree (IEEE 802.1 Bridge component)

module: ieee802-dot1q-mstp-bridge

augment /dot1q:bridges/dot1q:bridge/dot1q:component/dot1q:bridge-mst:
   +--rw mst-config-id!
      |   +--rw format-selector?     int32
      |   +--rw configuration-name?  string
      |   +--rw revision-level?      uint32
      |   +--ro configuration-digest? binary
   +--rw bridge-mstp!
      +--rw max-hops?              int32
      +--ro ist-internal-root-path-cost? uint32
   +--rw mst* [mstid]
      +--rw mstid                 uint16
      +--rw port-id-priority?      dot1q-types:priority-type
      +--ro internal-root-path-cost? uint32
      +--ro root-port-number?      dot1q-types:port-number-type

The config / state parameters augmented using ieee802-dot1q-mstp.yang grouping.
module:ieee802-dot1q-mstp-bridge

augment /if:interfaces/if:interface/dot1q:bridge-port:
  +--rw port-mstp!
    +--rw mst* [mstid]
      | +--rw mstid
      | +--ro msti-port-state? enumeration
      | +--ro msti-port-role? enumeration
      | +--rw msti-bridge-id-priority? dot1q-types:priority-type
      | +--rw msti-internal-port-path-cost? uint32
      | +--ro msti-regional-root-id? uint32
    +--ro msti-internal-root-path-cost? uint32
    +--ro msti-designated-bridge-id? uint32
    +--ro msti-designated-port-id? uint32

The config/state parameters augmented using ieee802-dot1q-mstp.yang grouping.
IEEE 802.1Qdy current YANG definition

Can be reused in other SDOs like BBF

Base YANG module – RSTP protocol specific YANG definitions as YANG grouping

- YANG module "ieee802-dot1q-rstp.yang"
  - YANG “Grouping” (rstp-per-component-protocol-specific-data)
    - RSTP YANG configuration/state data which are per component specific.

- YANG module "ieee802-dot1q-rstp-bridge.yang"
  - YANG “Grouping” (rstp-per-component-protocol-specific-data)
    - RSTP YANG configuration/state data which are per component specific.

- YANG module "ieee802-dot1q-rstp-bridge.yang"
  - YANG “Grouping” (rstp-per-interface-protocol-specific-data)
    - RSTP YANG configuration/state data which are per interface specific.

IEEE 802.1 device specific YANG module which uses Base YANG module

IEEE 802.1 device specific augments YANG definition

- Per component RSTP augment
  - YANG module "ieee802-dot1q-rstp-bridge.yang"
    - augment "/dot1q:bridges/dot1q:bridge/dot1q:component"
      - uses rstp-per-component-protocol-specific-data

- Per interface RSTP augment
  - YANG module "ieee802-dot1q-rstp-bridge.yang"
    - augment "/if:interfaces/if:interface/dot1q:bridge-port"
      - uses rstp-per-interface-protocol-specific-data
IEEE 802.1 UML-like view

**P802.1Qdy decoupling proposal – RSTP YANG**

IEEE 802.1 Base Bridge model

New YANG data model for RSTP

Existing data model in IEEE Std 802.1Q, Extended with RSTP new YANG data

**ieee802-dot1q-rstp**

**grouping rstp-per-component-protocol-specific-data**

```
+++rw rstp!
++-rw force-protocol-version? enumeration
++-ro cist-bridge-id? uint64
++-rw cist-bridge-id-priority? dot1q-types:priority-type
....
}
```

**grouping rstp-per-interface-protocol-specific-data**

```
+++rw rstp!
++-ro cist-port-id? uint16
++-rw cist-port-priority? dot1q-types:priority-type
++-rw external-port-path-cost? uint32
....
}
```

**uses ieee802-dot1q-rstp**

**uses ieee802-dot1q-rstp-bridge**

Note: Only the YANG is restructured considering reusability. The resultant YANG tree/functionality is the same as that on page 19 of [802-1Qdy-d2-0.pdf](#).
BBF Layer-2 Access device: RSTP/MSTP Implementation - possible UML-like view

**RSTP/MSTP per interface configuration/state data view**

- `ietf-hardware.yang`
  - `hardware`
  - `Component(*)`:
    - `name (key)`
    - ...
  - `0..n`

- `ietf-system.yang`
  - `system`
  - `rstp`
    - `configuration and state data`
  - `mstp`
    - `configuration and state data`
    - `list of MSTI`
    - `list of forwarder`
  - `0..1`

- `ietf-interfaces.yang`
  - `Interface(*)`:
    - `name (key)`
    - ...
  - `0..n`
  - `interface`
    - `rstp`
      - `configuration and state data`
    - `mstp`
      - `configuration and state data`
    - `0..1`

**Forwarding view**: List of forwarders per MSTI, which are implicitly map list of VIDs to MSTI.

- `forwarder(*)`:
  - `name (key)`
  - `forwarding-database`
  - `forwarder-ports list(*)`
  - `leafref`
  - `0..n`

**BBF specific interface-type on top of IETF interface-type to configure VLAN tag**

- `bbf-sub-interface-tagging.yang`
- `bbf-if-type.yang`
- `bbf-l2-forwarding-forwarders.yang`
- `bbf-l2-forwarding-forwarding-databases.yang`

E.g. `ifType ethernetCsMaCd, ieee8023adLag...`
Some observations on configuration/data consistency

• By using the YANG definitions for MSTP specified/proposed across IEEE P802.1Q-2022-REV and P802.1Qdy, there exists possibilities of the resultant configuration being inconsistent.

• Value(s) for one (set of) YANG object(s) could differ from another (set of) YANG object(s) that are otherwise supposed to have the same value(s).

• Even if the values are same/aligned between the different YANG objects, it would still mean that there are redundant objects in the YANG tree, i.e. two or more XPATHS pointing to essentially the same configuration.

• Details are explained in subsequent slides.
Some observations on configuration/data consistency

Possible inconsistencies

In IEEE P802.1Q-2022-REV, a container “bridge-mst” is defined with an object “mstid” which is a leaflist (/bridges/bridge/component/bridge-mst/mstid)

In IEEE P802.1Q-2022-REV, a container “bridge-mst” is defined which has a list “fid-to-mstid” that also has an object mstid /bridges/bridge/component/bridge-mst/fid-to-mstid/mstid

In IEEE P802.1Q-2022-REV, a container “bridge-mst” is defined which has a list “fid-to-mstid-allocation” that also has an object mstid /bridges/bridge/component/bridge-mst/fid-to-mstid-allocation/mstid

In P802.1Qdy, another list for the mst instance is introduced (/bridges/bridge/component/bridge-mst/bridge-mstp/mst)

Since all the above objects refer to the same MST Instance ID, there could be a different set of values in each of these lists
Some observations on configuration/data consistency

Possible inconsistencies

<table>
<thead>
<tr>
<th>XPATHS</th>
<th>Example configured values (mstd)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(leaf-list) /bridges/bridge/component/bridge-mst/mstdid</td>
<td>{100, 101, 102, 103}</td>
</tr>
<tr>
<td>(list) /bridges/bridge/component/bridge-mst/fid-to-mstdid {fid, mstd}</td>
<td>{1, 100}, {2, 109}, {3, 103}</td>
</tr>
<tr>
<td>(list) /bridges/bridge/component/bridge-mst/fid-to-mstdid-allocation {fids, mstd}</td>
<td>{500, 110}, {501, 100}, {502, 111}</td>
</tr>
<tr>
<td>(list) /bridges/bridge/component/bridge-mst/bridge-mstp/mst {mstd, config/state parameters}</td>
<td>{100, 200}</td>
</tr>
</tbody>
</table>

Highlighted in [RED] are misconfigurations w.r.t /bridges/bridge/component/bridge-mst/mstd leaf-list configuration
Avoiding inconsistent configuration and/or redundant data may require modifying the YANG objects specified in IEEE Std 802.1Q-2022 under “/bridges/bridge/component/bridge-mst”.

Possible solution

• Would it be acceptable to remove the “mstid” leaf-list from IEEE Std 802.1Q?
• The list /bridges/bridge/component/bridge-mst/bridge-mstp/mst proposed as part of P802.1Qdy should be sufficient to configure the MST instances. This list would be the main list/configuration for the set of MST instances in each Bridge component.
• If there is a need to refer/specify the “mstid” anywhere else in the YANG tree, a “leafref” type could be used.
• E.g. In the two lists “fid-to-mstid” and “fid-to-mstid-allocation”, the definition of “leaf mstid” could be changed from “type dot1qtypes:mstid-type” to “type leafref”.
• Also, these two lists would need to be moved to the newly proposed IEEE 802.1 specific YANG module “ieee802-dot1q-mstp-bridge” (since these would need to be augmented to Bridge-component).
Some observations on configuration/data consistency

Possible solution

Module ieee802-dot1q-bridge

leaf-list:mstid{
  type dot1qtypes:mstid-type;
  description
  "The list of MSTID values that are currently supported by the Bridge";
}

Remove the “mstid” leaf-list from IEEE Std 802.1Q
Some observations on configuration/data consistency

Possible solution

### Module ieee802-dot1q-bridge

<table>
<thead>
<tr>
<th>List: fid-to-mstid</th>
</tr>
</thead>
<tbody>
<tr>
<td>key &quot;fid&quot;;</td>
</tr>
<tr>
<td>description</td>
</tr>
<tr>
<td>&quot;The FID to MSTID allocation table.&quot;;</td>
</tr>
<tr>
<td>reference</td>
</tr>
<tr>
<td>&quot;12.12.2 of IEEE Std 802.1Q&quot;;</td>
</tr>
<tr>
<td>leaf fid {</td>
</tr>
<tr>
<td>type uint32;</td>
</tr>
<tr>
<td>description</td>
</tr>
<tr>
<td>&quot;The Filtering Database identifier.&quot;;</td>
</tr>
<tr>
<td>reference</td>
</tr>
<tr>
<td>&quot;12.12.2 of IEEE Std 802.1Q&quot;;</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>leaf mstid {</td>
</tr>
<tr>
<td>type dot1qtypes:mstid-type;</td>
</tr>
<tr>
<td>description</td>
</tr>
<tr>
<td>&quot;The MSTID to which the FID is to be allocated.&quot;;</td>
</tr>
<tr>
<td>reference</td>
</tr>
<tr>
<td>&quot;12.12.2 of IEEE Std 802.1Q&quot;;</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

Change the type of "mstid" in list "fid-to-mstid"

### Module ieee802-dot1q-bridge

<table>
<thead>
<tr>
<th>List: fid-to-mstid</th>
</tr>
</thead>
<tbody>
<tr>
<td>key &quot;fid&quot;;</td>
</tr>
<tr>
<td>description</td>
</tr>
<tr>
<td>&quot;The FID to MSTID allocation table.&quot;;</td>
</tr>
<tr>
<td>reference</td>
</tr>
<tr>
<td>&quot;12.12.2 of IEEE Std 802.1Q&quot;;</td>
</tr>
<tr>
<td>leaf fid {</td>
</tr>
<tr>
<td>type uint32;</td>
</tr>
<tr>
<td>description</td>
</tr>
<tr>
<td>&quot;The Filtering Database identifier.&quot;;</td>
</tr>
<tr>
<td>reference</td>
</tr>
<tr>
<td>&quot;12.12.2 of IEEE Std 802.1Q&quot;;</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>leaf mstid {</td>
</tr>
<tr>
<td>type leafref {</td>
</tr>
<tr>
<td>path '../../.dot1q-mstp:bridge-mstp/dot1q-mstp:mst/dot1q-mstp:mstid';</td>
</tr>
<tr>
<td>description</td>
</tr>
<tr>
<td>&quot;The MSTID to which the FID is to be allocated.&quot;;</td>
</tr>
<tr>
<td>reference</td>
</tr>
<tr>
<td>&quot;12.12.2 of IEEE Std 802.1Q&quot;;</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>
Some observations on Configuration/Data consistency

Possible Solution

Module ieee802-dot1q-bridge

list fid-to-mstid-allocation {
  key "fids";
  description
    "The FID to MSTID allocation table";
  leaf fids {
    type dot1qtypes:vid-range-type;
    description
      "Range of FIDs.";
    reference
      "12.12.2 of IEEE Std 802.1Q";
  }
  leaf mstid {
    type dot1qtypes:mstid-type;
    description
      "The MSTID to which the FID is allocated.";
    reference
      "12.12.2 of IEEE Std 802.1Q";
  }
}

Change the type of “mstid” in list “fid-to-mstid-allocation”

Module ieee802-dot1q-bridge

list fid-to-mstid-allocation {
  key "fids";
  description
    "The FID to MSTID allocation table";
  leaf fids {
    type dot1qtypes:vid-range-type;
    description
      "Range of FIDs.";
    reference
      "12.12.2 of IEEE Std 802.1Q";
  }
  leaf mstid { type leafref { path '..:/..dot1q-mstp:bridge-mstp/dot1q-mstp/mst/dot1q-mstp/mstid'; } 
    description
      "The MSTID to which the FID is allocated.";
    reference
      "12.12.2 of IEEE Std 802.1Q";
  }
}
Example of an existing IEEE 802.1 CFM YANG approach preventing configuration/data inconsistency

Module ieee802-dot1q-cfm
container cfm {
    
    list maintenance-domain {
        key "md-id";
        
        list maintenance-association {
            key "ma-id";
            
            list maintenance-association-mep {
                key mep-id;
                
            }
        }
    }
}

list maintenance-group {
    key "maintenance-group-id";
    
    leaf md-id {
        type leafref { path '/cfm/maintenance-domain/md-id';
        }
    }
    
    leaf ma-id {
        type leafref {
            path '/cfm'
            + '/maintenance-domain[md-id = current()]/../md-id'/maintenance-association/ma-id';
        }
    }
    
    list mep {
        key "mep-id";
        leaf mep-id {
            type leafref {
                path '/cfm/maintenance-domain[md-id = current()]/../md-id]'
                + '/maintenance-association[ma-id = current()]/../ma-id]/maintenance-association-mep/mep-id';
            }
        }
        
    }
}

Ref: ieee802-dot1q-cfm.yang

The CFM list (/cfm/maintenance-group/mep/mep-id) for the local MEP configuration refers the main list (/cfm/maintenance-domain/maintenance-association/maintenance-association-mep/mep-id) using leafref YANG type.

By using YANG leafref type approach, configuration/data inconsistency problem avoided between the main list and the local MEP configuration list.

Similarly,

- the “md-id” leaf configuration in the “maintenance-group” also defined with leafref YANG type to the main list “/cfm/maintenance-domain/md-id”.

- the “ma-id” leaf configuration in the “maintenance-group” also defined with leafref YANG type to the main list “/cfm/maintenance-domain/maintenance-association/ma-id”.

leafref
Some observations on configuration/data consistency

Possible solution

• If the lists “fid-to-mstid” and “fid-to-mstid-allocation” are updated, then these need to be placed in the IEEE 802.1 specific YANG module that has been proposed “ieee802-dot1q-mstp-bridge”, since these are linked to Bridge-component that may not be used by other SDOs.

• We could also decide to not disturb the existing objects “leaf-list mstid”, list “fid-to-mstid” and list “fid-to-mstid-allocation” and leave it to equipment vendors or even service providers / operators to ensure that an actual device configuration does not have inconsistencies.

• But it may be preferable to have the consistency checks inherent in the YANG model.
P802.1Qdy – RSTP/MSTP resultant YANG files

**YANG modules definition and tree**

*File name:*

“dy-balraj-reusability-across-sdos-yang-0624-v01.zip”

**Scope:**

- De-coupling proposal for reusability

**Includes:**

- ieee802-dot1q-mstp.yang => MSTP Base module
- ieee802-dot1q-mstp-bridge.yang => MSTP IEEE 802.1 specific module
- ieee802-dot1q-mstp-bridge.tree => MSTP IEEE 802.1 specific module YANG tree
- ieee802-dot1q-rstp.yang => RSTP Base module
- ieee802-dot1q-rstp-bridge.yang => RSTP IEEE 802.1 specific module
- ieee802-dot1q-rstp-bridge.tree => RSTP IEEE 802.1 specific module YANG tree
- ieee802-dot1q-bridge.tree => Resultant IEEE 802.1 Bridge Base YANG tree

No change proposed for the IEEE Base YANG file “ieee802-dot1q-bridge.yang”. The “bridge-mst” container definition considered as such.

---

**YANG modules definition and tree**

*File name:*

“dy-balraj-reusability-across-sdos-yang-without-inconsistencies-0624.v01.zip”

**Scope:**

- De-coupling proposal for reusability
- Solution for avoiding configuration inconsistency

**Includes:**

- ieee802-dot1q-mstp.yang => MSTP Base module
- ieee802-dot1q-mstp-bridge.yang => MSTP IEEE 802.1 specific module
- ieee802-dot1q-mstp-bridge.tree => MSTP IEEE 802.1 specific module YANG tree
- ieee802-dot1q-rstp.yang => RSTP Base module
- ieee802-dot1q-rstp-bridge.yang => RSTP IEEE 802.1 specific module
- ieee802-dot1q-rstp-bridge.tree => RSTP IEEE 802.1 specific module YANG tree
- ieee802-dot1q-bridge.yang => IEEE 802.1 Bridge Base model.
- Container “bridge-mst” removed from this module and redefined in ieee802-dot1q-mstp-bridge.yang module.
- ieee802-dot1q-bridge.tree => Resultant IEEE 802.1 Bridge Base YANG tree
Backup slides
P802.1Qdy - existing YANG view

<table>
<thead>
<tr>
<th>component (name)</th>
<th>string name; // r-w</th>
</tr>
</thead>
<tbody>
<tr>
<td>rstp</td>
<td></td>
</tr>
<tr>
<td>enum</td>
<td>force-protocol-version; // r-w</td>
</tr>
<tr>
<td>uint64</td>
<td>cist-bridge-id; // r</td>
</tr>
<tr>
<td>priority-type</td>
<td>cist-bridge-id-priority; // r-w</td>
</tr>
<tr>
<td>uint64</td>
<td>cist-root-id; // r</td>
</tr>
<tr>
<td>uint32</td>
<td>external-root-path-cost; // r</td>
</tr>
<tr>
<td>port-number-type</td>
<td>cist-root-port-number; // r</td>
</tr>
<tr>
<td>uint8</td>
<td>max-age; // r</td>
</tr>
<tr>
<td>uint8</td>
<td>hello-time; // r</td>
</tr>
<tr>
<td>uint8</td>
<td>forward-delay; // r</td>
</tr>
<tr>
<td>uint8</td>
<td>bridge-max-age; // r-w</td>
</tr>
<tr>
<td>uint8</td>
<td>bridge-hello-time; // r</td>
</tr>
<tr>
<td>uint8</td>
<td>bridge-forward-delay; // r-w</td>
</tr>
<tr>
<td>int32</td>
<td>tx-hold-count; // r-w</td>
</tr>
<tr>
<td>int32</td>
<td>migrate-time; // r</td>
</tr>
<tr>
<td>int32</td>
<td>time-since-topology-change; // r</td>
</tr>
<tr>
<td>counter64</td>
<td>topology-change-count; // r</td>
</tr>
</tbody>
</table>

Ref: 802-1Qdy-d2-0.pdf, Page 19

These RSTP protocol specific configuration and state data directly augment "/bridges/bridge/component".

A non-IEEE 802.1 device cannot import this YANG module due to the direct augmentation.

### VLAN Bridge component and port nodes + RSTP nodes

<table>
<thead>
<tr>
<th>bridge-port</th>
<th>leafref bridge-name; // r-w</th>
</tr>
</thead>
<tbody>
<tr>
<td>leafref</td>
<td>component-name; // r-w</td>
</tr>
<tr>
<td>rstp</td>
<td></td>
</tr>
<tr>
<td>enum</td>
<td>cist-port-state; // r</td>
</tr>
<tr>
<td>enum</td>
<td>cist-port-role; // r</td>
</tr>
<tr>
<td>bool</td>
<td>restricted-role; // r</td>
</tr>
<tr>
<td>bool</td>
<td>restricted-tcn; // r</td>
</tr>
<tr>
<td>uint16</td>
<td>cist-port-id; // r</td>
</tr>
<tr>
<td>priority-type</td>
<td>cist-port-priority; // r</td>
</tr>
<tr>
<td>int32</td>
<td>external-port-path-cost; // r-w</td>
</tr>
<tr>
<td>uint32</td>
<td>cist-root-id; // r</td>
</tr>
<tr>
<td>int32</td>
<td>cist-external-path-cost; // r</td>
</tr>
<tr>
<td>uint32</td>
<td>designated-bridge-id; // r</td>
</tr>
<tr>
<td>binary</td>
<td>designated-port-id; // r</td>
</tr>
<tr>
<td>bool</td>
<td>port-protocol-migration-check; // r-w</td>
</tr>
<tr>
<td>bool</td>
<td>admin-edge-port; // r-w</td>
</tr>
<tr>
<td>bool</td>
<td>oper-edge-port; // r</td>
</tr>
<tr>
<td>bool</td>
<td>auto-edge-port; // r-w</td>
</tr>
<tr>
<td>bool</td>
<td>auto-isolate-port; // r</td>
</tr>
<tr>
<td>bool</td>
<td>isolate-port; // r</td>
</tr>
</tbody>
</table>

These RSTP protocol specific interface configuration and state data directly augment "/interface/interface/bridge-port".

A non-IEEE 802.1 device cannot import this YANG module due to the direct augmentation.
These MSTP protocol specific configuration and state data directly augment "/bridges/bridge/component". A non-IEEE 802.1 device cannot import this YANG module due to the direct augmentation.

These MSTP protocol specific interface configuration and state data directly augment "/interface/interface/bridge-port". A non-IEEE 802.1 device cannot import this YANG module due to the direct augmentation.
## ieee802-dot1q-mstp.yang

YANG “Grouping” - Group MSTP Protocol specific configurations and state data.

A grouping for MSTP protocol configuration and state data parameters which are component specific.

A grouping for MSTP protocol configuration and state data parameters which are interface specific.

## ieee802-dot1q-rstp.yang

YANG “Grouping” - Group RSTP Protocol specific configurations and state data.

A grouping for RSTP protocol configuration and state data parameters which are component specific.

A grouping for RSTP protocol configuration and state data parameters which are interface specific.

## ieee802-dot1q-rstp-bridge.yang

Augments IEEE 802.1 Bridge Component/Bridge-port with RSTP YANG definitions from ieee802-dot1q-rstp.yang.

## ieee802-dot1q-mstp-bridge.yang

Augments IEEE 802.1 Bridge Component/Bridge-port with MSTP YANG definitions from ieee802-dot1q-mstp.yang.
module ieee802-dot1q-mstp {

    ....
    ....

    grouping mstp-per-component-protocol-specific-data {

        description
        "Grouping for MSTP configuration and state data";
        ....
        ....

    } // End of grouping mstp-per-component-protocol-specific-data

    grouping mstp-per-interface-protocol-specific-data {

        description
        "Grouping for MSTP configuration and state data augment under the interface";
        ....
        ....

    } // End of grouping mstp-per-interface-protocol-specific-data

}
P802.1Qdy decoupling proposal – MSTP YANG

Augmenting Bridge component/Bridge port with MSTP YANG objects

```yang
module ieee802-dot1q-mstp-bridge {

  augment "/dot1q:bridges/dot1q:bridge/dot1q:component/dot1q:bridge-mst" {
    when "../dot1q-rstp:rstp";
    description
      "Augment RSTP-capable Bridge component with MSTP configuration and
       management.";
    reference
      "13.24, 13.25, and 13.26 of IEEE Std 802.1Q.";
    uses mstp:mstp-per-component-protocol-specific-data;
  }

  augment "/if:interfaces/if:interface/dot1q:bridge-port" {
    when "dot1q-rstp:rstp";
    description
      "Augment RSTP Bridge Port with MSTP configuration";
    reference
      "13.24, 13.25, and 13.27 of IEEE Std 802.1Q.";
    uses mstp:mstp-per-interface-protocol-specific-data;
  }
}
```
P802.1Qdy decoupling proposal – RSTP YANG

Resultant YANG tree (IEEE 802.1 Bridge component)

module: ieee802-dot1q-rstp-bridge

augment /dot1q:bridges/dot1q:bridge/dot1q:component:
    +-rw rstp!
        +-rw force-protocol-version?   enumeration
        +-ro cist-bridge-id?           uint64
        +-rw cist-bridge-id-priority?  dot1q-types:priority-type
        +-ro cist-root-id?             uint64
        +-ro external-root-path-cost?  uint32
        +-ro cist-root-port-number?    dot1q-types:port-number-type
        +-ro max-age?                  uint8
        +-ro hello-time?               rt-types:timer-value-seconds16
        +-ro forward-delay?            uint8
        +-rw bridge-max-age?           uint8
        +-ro bridge-hello-time?        uint8
        +-rw bridge-forward-delay?     uint8
        +-rw tx-hold-count?            int32
        +-ro migrate-time?             int32
        +-ro time-since-topology-change? uint32
        +-ro topology-change-count?    yang:counter64

The config / state parameters augmented from ieee802-dot1q-rstp.yang grouping.
P802.1Qdy decoupling proposal – RSTP YANG

Resultant YANG tree (IEEE 802.1 Bridge Port)

module: ieee802-dot1q-rstp-bridge

augment /if:interfaces/if:interface/dot1q:bridge-port:
  +--rw rstp!
    +--ro cist-port-state? enumeration
    +--ro cist-port-role? enumeration
    +--ro restricted-role? boolean
    +--ro restricted-tcn? boolean
    +--ro cist-port-id? uint16
    +--rw cist-port-priority? dot1q-types:priority-type
    +--rw external-port-path-cost? uint32
    +--ro cist-root-id? uint32
    +--ro cist-external-path-cost? uint32
    +--ro designated-bridge-id? uint32
    +--ro designated-port-id? binary
    +--rw port-protocol-migration-check? boolean
    +--rw admin-edge-port? boolean
    +--ro oper-edge-port? boolean
    +--rw auto-edge-port? boolean
    +--rw auto-isolate-port? boolean
    +--ro isolate-port? boolean

The config / state parameters augmented from ieee802-dot1q-rstp.yang grouping.
module ieee802-dot1q-rstp {
    ..... 
    ..... 
    grouping rstp-per-component-protocol-specific-data {
        description
        "Grouping for RSTP configuration and state data";
        ..... 
        ..... 
    } // End of grouping rstp-per-component-protocol-specific-data 

    grouping rstp-per-interface-protocol-specific-data {
        description
        "Grouping for RSTP configuration and state data augment under the interface";
        ..... 
        ..... 
    } // End of grouping rstp-per-interface-protocol-specific-data 
}
Augmenting Bridge component/Bridge port with RSTP YANG objects

module ieee802-dot1q-rstp-bridge {

  augment "/dot1q:bridges/dot1q:bridge/dot1q:component" {
    description
    "Augment RSTP configuration and state data.";
    reference
    "13.24, 13.25, and 13.26 of IEEE Std 802.1Q.";
    uses rstp:rstp-per-component-protocol-specific-data;
  }

  augment "/if:interfaces/if:interface/dot1q:bridge-port" {
    description
    "Augment Bridge Port with RSTP configuration";
    reference
    "13.24, 13.25, and 13.27 of IEEE Std 802.1Q.";
    uses rstp:rstp-per-interface-protocol-specific-data;
  }
}