Ethernet Encryption Device (EDE) configuration

Table of Contents

Introduction

This is a white paper examining how to accommodate EDE configuration with the current bridge models and the evolving MACsec and MAC Privacy configuration. Ultimately this work is for Project IEEE 802.1AEdk MAC Privacy protection however this aspect involves the YANG model for IEEE 802.1AE [1].

IEEE 802.1AE states that EDE-CS (EDE-CC, EDE-SS) configuration is modeled after the Provider Bridge Model see IEEE 802.1Q [2]. Instances of Provider Bridge documentation are in https://www.ieee802.org/1/files/public/docs2017/cp-mholness-YANG-instance-document-0317-v02.pdf [3][4]. The instance document outlines how XML can be used to configure bridge components however, the XML input is an approximation to the model only and the YANG Schema was validated by a static set of tools that do not have full coverage.

This document specified here improves on this by using Yanglint[5] to validate xml configuration data against the published provider bridge YANG schemas. This increases the confidence that the XML is syntactically correct but still requires interpretation that sufficient elements have been configured for the intended operational model. This also tests aspects of the YANG schema such as xpath conditional statements. (The published YANG schema for bridges contained some xpath errors that were exposed during this exercise. Once those were corrected the configuration could be tested Corrected ieee802-dot1-bridge and ieee-dot1q-pb files have been submitted to maintenance to fix xpath issues). Using Yanglint the examples are more consistent, and the validity of the YANG models is improved.

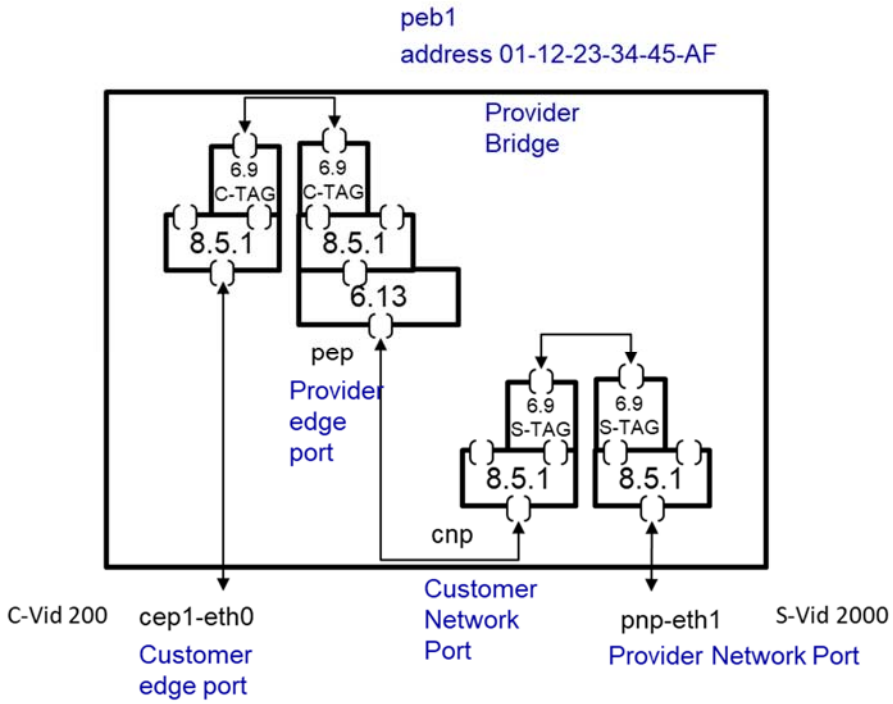

Figure 1 Provider Bridge Model

Updating the Provider Bridge configuration

This document uses YANG schemas that are published and, under development, to validate a provider bridge configuration and analyze it for applicability to Ethernet Data Encryption (EDE) devices.

The output of a Yanglint tested version of a provider bridge is shown below. Yanglint output is the data run against the YANG schema – when input equals to output the YANG is validated. (Note inspection of the input and the output is required since Yanglint will parse syntactically correct but unsupported XML silently). This general configuration model is also described in the instance document but here it is validated against the current YANG schema model. Some local assigned names have been changed for consistency and readability.

When running Yanglint the YANG modules must be loaded. Yanglint will auto load some files based on the dependencies in YANG module, but it is a good idea to load any tested components explicitly. The bridge model also contains features that must be enabled for the Yang model. In these examples all bridge features are enabled. (Note turning off and on features is a way test various xpath statements).

Following is the json output showing the yanglint commands.

```
~/ieeeYang$ yanglint
> load iana-if-type
> load ieee802-dot1q-bridge
> load ieee802-dot1q-pb
> feature ieee802-dot1q-bridge --enable *
> data -t config -f json provider-bridge1.xml
{
  "ieee802-dot1q-bridge:bridges": {
    "bridge": [
      {
        "name": "peb1",
```

```
        "address": "01-12-23-34-45-AF",
        "bridge-type": "provider-edge-bridge",
        "component": [
          {
            "name": "cvlan1",
            "type": "c-vlan-component"
          },
          {
            "name": "svlan2000",
            "type": "s-vlan-component"
          }
        ]
      }
    ]
  },
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "cep1-eth0",
        "type": "iana-if-type:ethernetCsmacd",
        "ieee802-dot1q-bridge:bridge-port": {
          "bridge-name": "peb1",
          "component-name": "cvlan1",
          "port-type": "customer-edge-port",
          "ieee802-dot1q-pb:cvid-registration": [
            {
              "cvid": 200,
              "svid": 2000
            }
          ]
        }
      },
      {
        "name": "pep1",
        "type": "iana-if-type:ethernetCsmacd",
        "ieee802-dot1q-bridge:bridge-port": {
          "bridge-name": "peb1",
          "component-name": "cvlan1",
          "port-type": "provider-edge-port"
        }
      },
      {
        "name": "cnp1",
        "type": "iana-if-type:ethernetCsmacd",
        "ieee802-dot1q-bridge:bridge-port": {
          "bridge-name": "peb1",
          "component-name": "svlan2000",
          "port-type": "customer-network-port"
        }
      },
      {
        "name": "pnp1-eth1",
        "type": "iana-if-type:ethernetCsmacd",
        "ieee802-dot1q-bridge:bridge-port": {
          "bridge-name": "peb1",
          "component-name": "svlan2000",
          "port-type": "provider-network-port"
        }
      }
    ]
  }
}
```


Explanation of Provider Bridge Configuration

The Example above is a Bridge with a C-VLAN and S-VLAN component. The instance document talks about autocreation of several of the components. For this example, since it is XML input and not real Netconf, we create all components explicitly – there is no auto creation capability (although it is still valid to summarize what can be autoconfigured). Figure 1 is provided for a reference of the provider bridge that corresponds to the configuration.

Note *The current ieee802-dot1q-bridge YANG does not include a Provider Edge port (pep) identity type. It has been added to the models here. Later this type is required for attaching the MacSec components.*

The data schema above defines a bridge with a C-VLAN and an S-VLAN component. The customer edge port (cep) labeled "cep-eth0" (a physical interface) is related to the "cvlan1" component. A cvid registration table is used to relate the cep "cep1" to a cvid "200" and a svid "2000" – identifies a pep. This inherently maps it to Customer Network Port (cnp) of the S-VLAN component. The svid 2000 auto creates the svlan2000 and an associated cnp. It has been pointed out in discussions that the registration table only allows a single cvid to a single svid at a time. Functionally this mapping works and replicates the MIB behavior. Multiple civds can refer to the same svid so the configuration can be used to allow many cvids to a single svid. In the registration table proposed later an alternate organization allows a more compact representation when mapping multiple inner VIDs (i.e. C-VIDs to S-VIDs)

Issue: The actual location of the pep and cnp is not specified. In YANG model an interface type is required as a reference pointer. The pep is required. The cnp does not necessarily need to be touched by configuration. In the xml models tested with Yanglint additional interfaces are supplied to satisfy the model. Documentation refers to virtual interfaces/ports however the creation of these is not documented. The pep interface(or port) is required to place various MACsec and MAC privacy configuration.

The example configuration enables a provider bridge with a frame that has an outer svid = 2000 and an inner cvid = 200 as illustrated in Figure 2. A pnp is created under a physical interface and associated with the S-VLAN component.

Figure 2 Provider Bridge MAC Frame

Creating an EDE-CS from Provider Bridge Configuration

The Provider Bridge model can be applied to an EDE-CS as described in IEEE802.1AE [1]. Note the Provider Bridge model is providing an infrastructure for the MACsec[1] and PAE[6] shims and the VLAN tag operation for the EDE. Historically MACsec and PAE were added to this model in 802.1AE-2018 and MAC Privacy Protection is being added as well in project 802.1AEdk. As mentioned earlier, the Provider Edge Port (pep), see Figure 3, is the logical place to put the PAE, MACsec and MAC Privacy Configuration and this port type had to be created as an port-type identity.

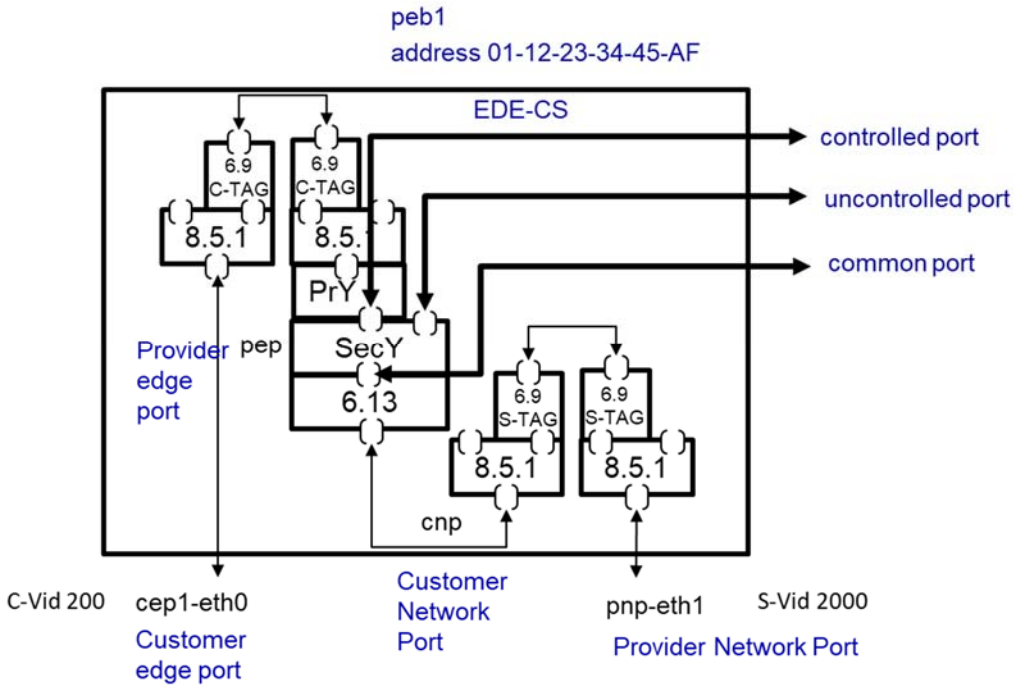

Figure 3 EDE-CS

The PAE, MACsec and MAC privacy components are placed on the pep interface as illustrated in

. Adding this to the model we can augment the interface (only the expansion of pep is shown).

```
{
   "name": "pep1",
   "type": "iana-if-type:ethernetCsmacd",
   "ieee802-dot1q-bridge:bridge-port": {
     "bridge-name": "peb1",
     "component-name": "cvlan1",
     "port-type": "provider-edge-port"
   },
   "ieee802-dot1ae:secy": {
     "controlled-port-number": 1,
     "verification": {
       "validate-frames": "strict",
       "replay-protect": true
     },
     "generation": {
       "max-transmit-channels": 16,
       "max-transmit-keys": 16,
       "protect-frames": true,
       "always-include-sci": true,
       "use-es": true,
       "use-scb": true,
       "user-priority-0": {
         "traffic-class": 1,
         "access-class-de0": 1,
         "access-class-de1": 1
       },
       "user-priority-1": {
         "traffic-class": 2,
         "access-class-de0": 2,
```

```
            "access-class-de1": 2
          },
          "user-priority-2": {
            "traffic-class": 4,
            "access-class-de0": 4,
            "access-class-de1": 4
          },
          "user-priority-3": {
            "traffic-class": 4,
            "access-class-de0": 4,
            "access-class-de1": 4
          }
        }
      },
```

Figure 4EDE-CS MACsec Frame Header

This results in a frame as illustrated in Figure 4. The only difference over the Provider Bridge model is the SecTag

According to the instance document the actual configuration is less than that illustrated because components are auto created. The pep interface identity type has been added to the bridge model and as a reference point to position for the Port authentication entity (PAE), MacSec (SecY) and Mac Privacy (PrY). A minimum configuration would remove the customer network port (cnp) and the input could look like this:

```
{
  "ieee802-dot1q-bridge:bridges": {
    "bridge": [
      {
        "name": "peb1",
        "address": "01-12-23-34-45-AF",
        "bridge-type": "provider-edge-bridge",
        "component": [
          {
            "name": "cvlan1",
            "type": "c-vlan-component"
          }
        ]
      }
    ]
  },
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "cep1-eth0",
        "type": "iana-if-type:ethernetCsmacd",
        "ieee802-dot1q-bridge:bridge-port": {
          "bridge-name": "peb1",
          "component-name": "cvlan1",
          "port-type": "customer-edge-port",
          "ieee802-dot1q-pb:cvid-registration": [
            {
              "cvid": 200,
              "svid": 2000
            }
```

```
          ]
        }
      }
      {
        "name": "pep1",
        "type": "iana-if-type:ethernetCsmacd",
        "ieee802-dot1q-bridge:bridge-port": {
          "bridge-name": "peb1",
          "component-name": "cvlan1",
          "port-type": "provider-edge-port"
        },
        "ieee802-dot1ae:secy": {
          "controlled-port-number": 1,
          "verification": {
            "validate-frames": "strict",
            "replay-protect": true
          },
          "generation": {
            "max-transmit-channels": 16,
            "max-transmit-keys": 16,
            "protect-frames": true,
            "always-include-sci": true,
            "use-es": true,
            "use-scb": true
            }
          }
        }
      },
        "ieee802-dot1x:pae": {
          "pae-system": "pae1"
        }
      },
      {
        "name": "pnp1-eth1",
        "type": "iana-if-type:ethernetCsmacd",
        "ieee802-dot1q-bridge:bridge-port": {
          "bridge-name": "peb1",
          "component-name": "svlan2000",
          "port-type": "provider-network-port"
        },
        "ieee802-dot1x:pae": {
    ]
  },
  "ietf-system:system": {
    "contact": "test",
    "ieee802-dot1x:pae-system": {
      "name": "pae1",
      "system-access-control": "enabled"
    }
  }
}
```


Essentially the cvid(s) to svid relationship and the pep to cnp component is auto created from the c-vid registration table.

When it comes to the other type of EDEs (EDE-CC, EDE-SS) there is currently no semantics that fit this model. The c-vid to s-vid relationship is automatic but another type of table would be needed.

The cvid-registration table is illustrated here:

C-VID Registration Table	Description
Cvid value	The value of the Customer VLAN id on the Customer edge port. (Table key)
Svid Value	The S-VLAN tag. Auto creates an S-VLAN component and the CNP and PNP and links the PEP of the C-VLAN component to the CNP.
Untagged-pep	A boolean indicating frames for this C-VLAN should be forwarded untagged through the Provider Edge Port.
Untagged-cep	A boolean indicating frames for this C-VLAN should be forwarded untagged through the Customer Edge Port.

Table 1 C-VID registration table

This table is indexed by an inner C-VID and references an outer S-VID. It also is intended to auto create the pep and cnp. There is an implied linkage from the C-VLAN component to the S-VLAN component through the SVID value. There are also flags to allow untagged cases.

EDE-CC and EDE-SS Configuration Requirements

While the Provider Bridge Model provides a framework for creating an EDE-CS the relationships of the model are hard wired for customer VLANs and service VLANs. Currently, with the published YANG it does not lend itself to changing the tag types to two C-VLAN components or two S-VLAN components. (A few attempts at resolving this have been made – this document contains a current proposal). The assumption is that a cep is under a C-VLAN component. The relationship is configured in the registration table (Table 1) since a cep can be related to an S-VLAN component through an svid. The registration table uses a cvid as a key. When the customer side component is an S-VLAN component the table is not intuitive because the underlying assumption is the customer tag is a C-TAG . To remedy this, we need a table where the TAG type can be specified as well as the tag values or the TAG values can apply to any type of component. It turns out that since the table is under a leaf that is refencing a component with a implied TAG type, simply changing the names in the table allows a logical mapping.

Naming alternatives. Here are the current names. Note these are YANG identities only. Creating alternative identities is a trivial change to existing YANG. (Identities can be provided to add clarity).

Current Identities

- Customer Edge port (cep) – Must create - Implies a C-VID tagged edge.
- Provider Edge port (pep) – auto created
- Customer Network port (cnp)– auto created
- Provider Network port (pnp)– Must create - Implies an S-VID tagged edge.

For EDEs suggest identities:

- new identity port type - red-side-port – The ethernet unencrypted/non-private side
- pep – auto created – leave name for now
- cnp – auto created – leave name for now – this component can be hidden.
- new identity port type - black-side-port – The ethernet encrypted/private side.

EDE tag registration table	Description
Black-side-vid	VID value – type (TPID) depends on linked component type
Red-side-vid	VID range – TPID depends on linked component type
Untagged-pep	A boolean indicating frames for this VLAN should be forwarded untagged through the Provider Edge Port.
Untagged-red-side-port	A boolean indicating frames for this VLAN should be forwarded untagged through the Red-side Port.

Table 2 EDE Registration table

One way to do this discussed in the 802.1 Security group was to use an inner and outer port designation to get away from the customer/provider aspects. In this document a smaller change is proposed – creating only Red-side-port and Black-side-port alternatives for port types.

Note- The suggestion of reversing the table index from the Red-side to the Black-side allows a more compact format for the table by allowing the Red-side to be a VLAN range. In YANG , strictly speaking, the table index can also be a range, but this could lead to ambiguous entries (ranges are string values in YANG input and less easy to check for conflicts at data entry on input). The above organization is merely an optimization in data entry over the other table organization but allow for ranges and more compact representation of mapping.

Referring to the sample config below with the table and the new identities. The local names red1 and black1 are used for clarification but they could be called anything. The type of component is still C-VLAN or S-VLAN and this is important because it will determine the VID TPID type.

Here is a sample config .

Note both “red1” and “black1” are c-vlan components – implying C-TAGs. And later, the components are associated to the VID values in the ede-tag-registration table.

```
~/ieeeYang$ $ yanglint
> load iana-if-type
> load ieee802-dot1q-bridge
> load ieee802-dot1q-pb
> load ieee802-dot1x
> load ieee802-dot1ae
> load ieee802-dot1ae-pry
> feature ieee802-dot1q-bridge --enable *
> data -t config -f json provider-bridge1-ede-cc1.xml
{
  "ieee802-dot1q-bridge:bridges": {
    "bridge": [
      {
        "name": "ede1",
        "address": "01-12-23-34-45-AF",
        "bridge-type": "provider-edge-bridge",
        "component": [
          {
            "name": "red1",
            "type": "c-vlan-component"
          },
```

```
          {
            "name": "black1",
            "type": "c-vlan-component"
          }
        ]
      }
    ]
  },
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "red-side-eth0",
        "type": "iana-if-type:ethernetCsmacd",
        "ieee802-dot1q-bridge:bridge-port": {
          "bridge-name": "ede1",
          "component-name": "red1",
          "port-type": "red-side-port",
          "ieee802-dot1ae:ede-tag-registration": [
            {
              "black-side-vid": 2000,
              "red-side-vids": "200-205"
            }
          ]
        },
        "ieee802-dot1x:pae": {

        }
      },
      {
        "name": "vp1",
        "type": "iana-if-type:ethernetCsmacd",
        "ieee802-dot1q-bridge:bridge-port": {
          "bridge-name": "ede1",
          "component-name": "red1",
          "port-type": "provider-edge-port"
        },
        "ieee802-dot1ae:secy": {
          "controlled-port-number": 1,
          "verification": {
            "validate-frames": "strict",
            "replay-protect": true
          },
          "generation": {
            "max-transmit-channels": 16,
            "max-transmit-keys": 16,
            "protect-frames": true,
            "always-include-sci": true,
            "use-es": true,
            "use-scb": true,
            "user-priority-0": {
              "traffic-class": 1,
              "access-class-de0": 1,
              "access-class-de1": 1
            },
            "user-priority-1": {
              "traffic-class": 2,
              "access-class-de0": 2,
              "access-class-de1": 2
            },
            "user-priority-2": {
              "traffic-class": 4,
              "access-class-de0": 4,
              "access-class-de1": 4
            },
            "user-priority-3": {
              "traffic-class": 4,
```

```
              "access-class-de0": 4,
              "access-class-de1": 4
            }
          }
        },
        "ieee802-dot1x:pae": {
          "pae-system": "pae1"
        }
      },
      {
        "name": "black-side-eth1",
        "type": "iana-if-type:ethernetCsmacd",
        "ieee802-dot1q-bridge:bridge-port": {
          "bridge-name": "ede1",
          "component-name": "black1",
          "port-type": "black-side-port"
        },
        "ieee802-dot1x:pae": {

        }
      }
    ]
  },
  "ietf-system:system": {
    "contact": "test",
    "ieee802-dot1x:pae-system": {
      "name": "pae1",
      "system-access-control": "enabled"
    }
  }
}
>
```


An equivalent table would need to be created for SNMP (there is already a C-VID registration table).

To Summarize this configuration example has:

A bridge with two components

- red1 – a C-VLAN component
- black1 – a C-VLAN component

Two Physical ports

- eth0 – a red-side-port - associated with red1 component
- eth1 – a black-side-port associated with black1 component

Two Virtual ports

- pep – where pae, secy, and pry are attached – associated with the red1 component
  - Most attributes related to the MACsec and PAE and PrY are here.
- cnp – no specific configuration but associates the pep to the second component and the second tag operations.

The ede-registration table that maps VID values.

- black-side-vid - One VID value Type determined by linkage of black-side-port to the component.
- Red-side-vid – One or more VID values (type determined by linkage of the re-side-port to the component.)

EDE-M Configuration

EDE-M is configured on a simple VLAN bridge model with a single TAG. Strictly speaking EDE-M works without any change to the existing bridge models. However, it is illustrated here making use of the new red-side-port and black-side-port types for readability.

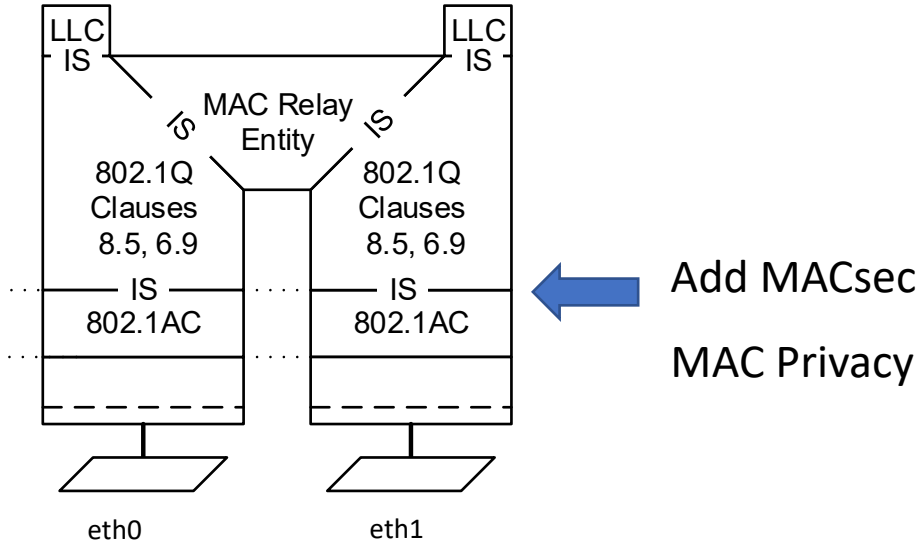

Figure 5 Basic VLAN Unaware Bridge

A basic VLAN Unaware bridge has the following configuration format.

```
> data -t config -f json basic-bridge.xml
{
  "ieee802-dot1q-bridge:bridges": {
    "bridge": [
      {
        "name": "bridge1",
        "address": "10-10-10-10-10-10",
        "bridge-type": "customer-vlan-bridge",
        "component": [
          {
            "name": "d-relay1",
            "id": 1,
            "type": "d-bridge-component"
          }
        ]
      }
    ]
  },
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth0",
        "type": "iana-if-type:bridge",
        "ieee802-dot1q-bridge:bridge-port": {
          "bridge-name": "bridge1",
          "component-name": "d-relay1",
          "port-type": "d-bridge-port"
        }
      },
      {
```

```
        "name": "eth1",
        "type": "iana-if-type:ethernetCsmacd",
        "ieee802-dot1q-bridge:bridge-port": {
          "bridge-name": "bridge1",
          "component-name": "d-relay1",
          "port-type": "d-bridge-port"
        }
      }
    ]
  }
}
```

Note that Secy, PrY and PAE are also tag agnostic so all that is needed is to place the MACsec and MAC Privacy components under the correct component as before.

What is required is a port with an interface that can be used to place the MACsec / MAC Privacy config as before. Below the red-side-port and black-side-port replace the d-bridge-port type.

The configuration for EDE-M.

```
> data -t config -f json basic-bridge.xml
{
  "ieee802-dot1q-bridge:bridges": {
    "bridge": [
      {
        "name": "bridge1",
        "address": "10-10-10-10-10-10",
        "bridge-type": "customer-vlan-bridge",
        "component": [
          {
            "name": "d-relay1",
            "id": 1,
            "type": "d-bridge-component"
          }
        ]
      }
    ]
  },
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth0-red-side",
        "type": "iana-if-type:bridge",
        "ieee802-dot1q-bridge:bridge-port": {
          "bridge-name": "bridge1",
          "component-name": "d-relay1",
          "port-type": "red-side-port"
        },
        "ieee802-dot1x:pae": {

        }
      },
      {
        "name": "eth1-black-side",
        "type": "iana-if-type:ethernetCsmacd",
        "ieee802-dot1ae-pry:pry": {
          "mac-privacy": "enabled",
          "user-priority-to-pry": [
            {
              "user-priority": 0,
              "privacy-type": "frame-a"
            },
            {
```

```
            "user-priority": 1,
            "privacy-type": "express-channel"
          },
          {
            "user-priority": 2,
            "privacy-type": "express-channel"
          },
          {
            "user-priority": 3,
            "privacy-type": "express-channel"
          },
          {
            "user-priority": 4,
            "privacy-type": "standard-channel"
          },
          {
            "user-priority": 5,
            "privacy-type": "standard-channel"
          },
          {
            "user-priority": 6,
            "privacy-type": "standard-channel"
          },
          {
            "user-priority": 7,
            "privacy-type": "standard-channel"
          }
        ],
        "privacy-channel": [
          {
            "pc": "standard-channel",
            "max-per-second-bitrate": "10000000000",
            "max-mppdu-size": 1500,
            "mppdu-priority": 3
          }
        ]
      },
      "ieee802-dot1q-bridge:bridge-port": {
        "bridge-name": "bridge1",
        "component-name": "d-relay1",
        "port-type": "black-side-port"
      },
      "ieee802-dot1ae:secy": {
        "controlled-port-number": 1,
        "verification": {
          "validate-frames": "strict",
          "replay-protect": true
        },
        "generation": {
          "max-transmit-channels": 16,
          "max-transmit-keys": 16,
          "protect-frames": true,
          "always-include-sci": true,
          "use-es": true,
          "use-scb": true
        }
      },
      "ieee802-dot1x:pae": {
        "pae-system": "pae1"
      }
    }
  ]
},
"ietf-system:system": {
  "contact": "test",
  "ieee802-dot1x:pae-system": {
```

```
      "name": "pae1",
      "system-access-control": "enabled"
    }
  }
}
```

Summary

This paper has shown that with the addition of a two port-types – designated red-side and black-side and a new EDE registration table existing bridge configuration can handle all the EDE variants. Also, Yanglint has been used to validate the new and existing YANG schemas improving the validity and the completeness of the models.

References:

1. IEEE Std 802.1AE™, IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Security.
2. IEEE Std 802.1Q™, IEEE Standard for Local and Metropolitan Area Networks: Bridges and Bridged Networks.
3. IEEE Std 802.1Qcp, IEEE Standard for Local and Metropolitan Area Networks: Bridges and Bridged Networks—Amendment: YANG Data Model
4. https://www.ieee802.org/1/files/public/docs2017/cp-mholness-YANG-instance-document-0317-v02.pdf.
5. https://github.com/CESNET/libyang
6. IEEE Std 802.1X™, IEEE Standard for Local and Metropolitan Area Networks: Port-Based Network Access Control.