Comment on ieee802dot1CBdb.yang MASK and Match

Don Fedyk (dfedyk@labn.net)

Currently YANG MAC address mask and match

```
leaf destination-mac-mask {
      type uint48;
      description
        "Specifies a 48-bit mask. A bitwise AND operation is performed
         between destination-mac-mask and the
         destination_address_parameter passed by the ISS indication
         primitive to the Mask-and-match Stream identification
         function. The resulting 48-bit information is the masked
         destination_address that is used as input for the instance of
         the Mask-and-match Stream identification function. If
         destination-mac-mask has a value of 0, the destination-address
         parameter is ignored.";
      reference
        "Clause 9.1.6.1 of IEEE Std 802.1CBdb";
            }
    leaf destination-mac-match {
      when '../destination-mac-mask';
      type uint48;
      description
        "Specifies the 48-bit value of the masked destination_address,
         to be matched by the instance of the Mask-and-match Stream
         identification function.";
      reference
        "Clause 9.1.6.2 of IEEE Std 802.1CBdb";
    }
```

My Understanding

- Two integers Representing a MAC address and a value of care/ don't care bits. (bitwise and)
- This is very user-unfriendly. (It is OK if you are doing this with a machine but as far as humans go - not great.)
- My original thought was why not represent these as hexadecimal in mac address format?
  - At least then it would be easier to read.
  - The problem is that you still need two numbers because you cannot specify masking in one entry in hexadecimal.
- Then it occurred to me that YANG can do both operations in one entry in binary.
  - Disclaimer I'm not a fan of YANG Strings. The regex is harder to code and easy to get wrong, But there are cases where it can work.
- Since you can define a regular expression (regex) you can create a single bitmap entry.
- Managed objects stay the same

There are two of these one for source and one for destination

An Alternative

```
leaf dst-mac-mask-match {
      type string {
            pattern '[01*]{8}([-][01*]{8}){5}';
      }
      description
         “A bit pattern with masked bits set to
don’t care. Exact match is specified by setting
all bits";
    }
```

- This string pattern allows the same operation in one entry and it is readable
- "dst-mac-mask-match": "00010010-00000100-011100*0-01100000-00000000-0100****",
- Regex simply says a string of 6, 8-bit numbers separated by dashes where the values are (0 or 1 or *) * means don’t care.
- Comparing that to the current encoding:
- "destination-mac-mask": "281474943156208",   // Hex fffffdfffff0
- "destination-mac-match": "19810274508870",   // HEX 120470600046

Less Radical

```
leaf destination-mac-mask {
      type ieee:mac-address;
      description
        "Specifies a 48-bit mask. A bitwise AND operation is performed
         between destination-mac-mask and the
         destination_address_parameter passed by the ISS indication
         primitive to the Mask-and-match Stream identification
         function. The resulting 48-bit information is the masked
         destination_address that is used as input for the instance of
         the Mask-and-match Stream identification function. If
         destination-mac-mask has a value of 0, the destination-address
         parameter is ignored.";
      reference
        "Clause 9.1.6.1 of IEEE Std 802.1CBdb";
                }
    leaf destination-mac-match {
      when '../destination-mac-mask';
      type ieee:mac-address;
      description
        "Specifies the 48-bit value of the masked destination_address,
         to be matched by the instance of the Mask-and-match Stream
         identification function.";
      reference
        "Clause 9.1.6.2 of IEEE Std 802.1CBdb";
    }
```

Using Hex from Mac-address format

"destination-mac-mask": “ff-ff-fd-ff-ff-f0”

"destination-mac-match": "12-04-70-60-00-46”

Conclusion

- It comes down to what you want to accomplish:
  1. Function – all forms do the equivalent operation
  2. Readability – Hex (or binary) versus integer is preferable
  3. Error prone – The single binary entry ensures no mismatch.
  4. Compact – Hex (or integer (see 2)) is more compact

  - I recommend Hex or Binary but not integer.

Another Possibility - Liberal string

```
leaf dst-mac-mask-match {
      type string {
              pattern '([Mm][Aa][Cc][\s=]*)*\s*[0-9a-f]{2}([-][0-9a-f]{2}){5}
                      (\s*([Mm][Aa][Ss][Kk][\s=]*)*\s*[0-9a-f]{2}([-][0-9a-f]{2}){5}){0,1}';
      }
      description
          "A Hex MAC address in IEEE MAC address format
           followed by an optional Hex Mask in IEEE MAC
           address format
           [MAC =] xx-xx-xx-xx-xx-xx [[Mask =] yy-yy-yy-yy-yy-yy]";
    }
```

dst-mac-mask-match: 12-04-70-60-00-46 mask ff-ff-fd-ff-ff-f0

Or exact match:

dst-mac-mask-match: 12-04-70-60-00-46

It is liberal because Labels and spaces and equal signs are optional

: MAC = 12-04-70-60-00-46 Mask = ff-ff-fd-ff-ff-f0 or

: 12-04-70-60-00-46 ff-ff-fd-ff-ff-f0

Optional label MAC,

followed by optional =

followed by mandatory MAC address Hex format,

followed by optional label Mask,

followed by optional =

followed by 1 optional Mask in MAC address Hex format

NOT Recommended

https://yangcatalog.org/yangre/