MAC Address Format
Summary and Suggestion
Updated Presentation

Scott Mansfield
Ericsson
Summary

• This is a discussion as old as time...
  – Well... July 14, 2008 with the first draft in the IETF of what would become RFC 6091 (Common YANG Data Types) which would become RFC 6991

• Subsequently...
  – IEEE 802.1Qcp in an early draft in 2017 provided a different format and was published in 2018
  – IEEE 802.3 in IEEE 802.3.2-2019 Used the IETF definition of mac-address from RFC 6991
MAC Address Format

• IETF and IEEE 802.1 have different patterns for mac-address
  – IETF Format: pattern '[0-9a-fA-F]{2}(:[0-9a-fA-F]{2}){5}';
    • uses ‘:’ as separator
  – IEEE 802.1 Format: pattern "[0-9a-fA-F]{2}([-0-9a-fA-F]{2}){5}";
    • uses ‘-’ as separator
    • Also ‘:’ has a different interpretation in IEEE 802 specs
      than ‘-’ does. The ‘:’ indicates bit-reversal of each hex digit.
        – However the bit-reversal usage is historic
          » An amendment to IEEE Std 802-2014 is needed to reflect
             that officially
Not just a ‘-’ or ‘:’ problem

- IEEE definition
- `ieee802-types.yang`
- “-”

- Pattern allows upper and lower case characters but description says uppercase is used.

- IETF definition
- `ietf-yang-types.yang`
- “:”

- Pattern allows upper and lower case but notes that lower case is canonical.
Issue with strings

• mac-address typedef is a string in YANG

• That means when mac-address is used as a key, the input format used must match not only the separator (‘:’ or ‘-’) but the case of the characters representing the hexadecimal number

• If a mac-address is used as a key, or if two mac-addresses need to be compared, a normalized format would be useful.
Some Example Trouble Spots

- ietf-l2vpn-svc uses mac-address as a key
- ietf-i2rs-rib.yang has a mac-address leaf that the description says is “used for matching”
- ieee802-dot1q-lldp.yang uses mac-address as a key
- ieee802-dot1q-tnsn-types.yang defines a mac-address without using the datatype, but uses the same pattern as the ieee mac-address datatype
Why SNMP is different

• In SNMP a MacAddress is an OCTET STRING of size 6 with a display hint.
• On the wire the MacAddress is treated as a string of octets that are not affected by the display hint or the separator used.
• So AE-12-FF would be the same as ae:12:ff
What to do

• Common wisdom says it is too late to change either the IEEE or IETF definition to use a 6 byte binary array
  – This would fix the “on-the-wire” and key comparison issue
  – Whatever is done should be done for any OUI types also
  – Another concern is that the current patterns only support 48-bit MAC addresses, but IEEE Std 802-2014 also mentions “extended address” or 64-bit MAC addresses.

• Identify potential conflicts
  – Modules that use both yang:mac-address and ieee:mac-address and try to compare them or present two different input formats because of the pattern differences.
  – Even if only one definition is used, some hints or guidelines should be created because the format of the string (upper/lower case) matters for comparison

• A Suggestion is provided on the next slide
• Followed by a summary of various options on how to proceed
Suggestion

• Leave the IETF and IEEE definitions alone
• Create a new datatype in ieee802-types.yang
  – Implementations could use the normalized format when mac-address is used as a key or there is a concern over the string matching

```yang
typedef mac-address-normalized {
  type string {
    pattern "[0-9A-F]{2}([0-9A-F]{2}){5}";
  }
  description "The mac-address type represents a normalized MAC address format. There is no ambiguity in the format so string comparison is possible.";
  reference "3.1 of IEEE Std 802-2014
  8.1 of IEEE Std 802-2014
  IETF RFC 6991";
}
```

The pattern has no separator and allows only upper case, this avoids any ambiguity.
Summary

• Do Nothing

• Have 802.1 YANG use RFC 6991
  – The fact that mac-address is a string is still an issue because of how string comparison works

• Normalized mac-address format typedef
  – Suggestion from Previous Slide

• Other thoughts?
  – YANG support for display-hint like functionality
  – String equivalence pattern to provide flexibility for string (and key) comparisons
  – Other input and/or display capabilities