MAC Address Format
Summary and Suggestion

Scott Mansfield
Ericsson
MAC Address Format

• IETF and IEEE have different patterns for mac-address
  – IETF Format: pattern '\([0-9a-fA-F]{2}\)(:[0-9a-fA-F]{2}){5}';
    • uses ‘:’ as separator
  – IEEE Format: pattern "[0-9a-fA-F]{2}([-0-9a-fA-F]{2}){5}";
    • uses ‘-’ as separator
    • Also ‘:’ has a defined meaning in IEEE specs (bit-reversal of each hex digit)
    • However the bit-reversal issue is historic (but there really should be an amendment to official recognize that fact)
Not just a ‘-’ or ‘:’ problem

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

- IETF definition
  - Pattern allows upper and lower case but makes no indication on which is used.

```c
typedef mac-address {
  type string {
    pattern "[0-9a-fA-F]{2}([-0-9a-fA-F]{2})\{5\};
  }
  description
  "The mac-address type represents a MAC address in the canonical format and hexadecimal format specified by IEEE Std 802. The hexadecimal representation uses uppercase characters."
  reference
  "3.1 of IEEE Std 802-2014
  8.1 of IEEE Std 802-2014"
}
```
Issue with strings

- mac-address typedef is a string in YANG
- That means when mac-address is used as a key, the format used must match not only the separator (‘:’ or ‘-’) but the case of the character representing the hexadecimal number
- Review of implementations indicates that ‘:’ or ‘-’ doesn’t change the ordering of hexadecimal digits in the string.
  - However the issue of upper and lower case and the separator cause issues when comparing mac-addresses
Why SNMP is different

• In SNMP a MacAddress was 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

• Identify potential conflicts
  – Modules that use both yang:mac-address and ieee:mac-address and try to compare them
  – 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

• Suggestion: Normalized mac-address format typedef (next slide)
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

```plaintext
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 for 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";
}
```