# String comparison issue

mac-address example

Scott Mansfield
scott.mansfield@ericsson.com

# Introduction

- YANG has an issue with strings when strings are used as keys or simply when comparing strings.
- If a string has a pattern, it is possible to create a situation when equivalent strings are not considered equivalent.
- The example in this presentation is mac-address.
- Recommendation
  - Fix the underlying issue with YANG string comparisons.

# 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.
      - The bit-reversal usage is historic and common usage is that '-' and ':' are treated the same.
- However, today (in 2025)
  - AA-BB-CC-DD-EE-FF should be equivlant to aa:bb:cc:dd:ee:ff

# Not just a '-' or ':' problem

- IEEE definition
- ieee802-types.yang
- "-"
- Pattern allows upper and lower case characters but description says uppercase is used.

```
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
    hexidecimal representation uses uppercase characters.";
  reference
    "3.1 of IEEE Std 802-2014
    8.1 of IEEE Std 802-2014";
}
```

- IETF definition
- ietf-yang-types.yang
- ":"
- Pattern allows upper and lower case but notes that lower case is canonical.

```
typedef mac-address {
  type string {
    pattern '[0-9a-fA-F]{2}(:[0-9a-fA-F]{2}){5}';
  }
  description
   "The mac-address type represents an IEEE 802 MAC address.
    The canonical representation uses lowercase characters.

    In the value set and its semantics, this type is equivalent
    to the MacAddress textual convention of the SMIv2.";
  reference
   "IEEE 802: IEEE Standard for Local and Metropolitan Area
              Networks: Overview and Architecture
    RFC 2579: Textual Conventions for SMIv2";
}
```

# 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-tsn-types.yang defines a mac-address without using the datatype, but uses the same pattern as the ieee mac-address datatype
- ieee802-ethernet-link-oam and ieee802-ethernet-pon.yang use ieee mac-address datatype

# 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 mechanism to indicate equivalence would be useful.

# Options

- Create a new mac-address type and deprecate the mac-address types in ietf and ieee types files.
  - '[0-9A-F]{2}([0-9A-F]{2}){5}'
    - Remove seperator and choose Upper Case
    - Call this "normalized" and deprecate the pattern used in IETF and IEEE
  - Over time, this could solve the "string" problem, because there would be only one valid format for storing a mac-address. Painful for existing deployments and instance data.
  - However, display format is hard to read.
- Modify the mac-address patterns in both ietf and iee to be inclusive of everything.
  - '[0-9a-fA-F]{2}(:-[0-9a-fA-F]{2}){5}’
    - All patterns allowed
  - Issue: doesn't solve the string equivalance problem
- Add something to the YANG language that will indicate how strings are equivalent.
  - For example 'ignore-case' or 'options are equivalent' functions or statements.
  - This is backwards compatible and doesn't require instance data to change.
- Do what SNMP Did
  - Change the storage of mac-addresses to be 6 octets.
  - Have some kind of "display-hint" that will provide how the mac-address could be displayed.
- Do Nothing

# Summary

- Keep in mind this isn't just a mac-address problem.
- This is a problem anytime a permissive pattern is used for strings, and then the string is used as a key or expected to be compared against other strings from different modules.
- One option to partially fix this issue:
  - Create a normalized pattern for mac-address and migrate to that overtime (in both IETF and IEEE YANG modules).
- Two options to fix (generically for all strings, not just mac-addresses):
  - Provide a mechanism to indicate that a string is an 'id' or something that will be used for comparison and avoid the use of permissive patterns.
  - Create a YANG statement that will provide an indication of how strings can be considered the same. This is backwards compatible but would require updates to clients and servers to implement a new string comparison method.
- Elephant in the room
  - Neither IETF nor IEEE will find it easy to adopt the format from the other group.
    - My position is the best solution is to fix YANG to allow a string equivalence capability is added.