MAC Address Canonical

Don Fedyk <dfedyk@labn.net>

LabN Consulting L.L.C.

IETF and IEEE both define MAC address formats

- We have had discussion for years about the problem. Both are strings and both have differing syntax.
- IETF: 00:11:22:aa:BB:cc
- IEEE: 00-11-22-aa-BB-cc
- Strings are not values
  - Comparing is not guaranteed
  - Can't be used as indexes.
- While we are waiting for a fix that may never come it is possible to express our intent in current YANG with a small extension.
- This is the proposal to simply use what we have.

Rules for a solution

- MUST preserve the lexical representation of values.
- MUST provide deterministic comparison semantics.
- MUST be minimally invasive to existing YANG modules.
- MUST be opt-in and backward compatible.
- MUST be general across string-derived types.
- SHOULD be simple to implement.
- SHOULD avoid complex transformation languages.

A small extension with identities

- YANG allows extensions these extension extend the language.
- It documents the behavior, but it does not implement a solution.
- The argument attribute is a string.
- We can use an identity to define the string.
- Implementors can read the associated string
- Additional identities can be added.

```
extension canonical-form {
   argument form;
   description
     "Specifies a deterministic canonicalization form
      used to derive the canonical form of a value.";
 }

 identity canonical-form {
   description
     "Base identity for canonical forms.";
 }
 identity mac-48 {
   base canonical-form;
   description
     "Canonical form for 48-bit MAC addresses.";
 }
```

Usage One simple line

- This extension is usable with existing YANG tooling. Tools such as pyang can parse the extension and make its argument available as a string. Validation that the string identifies a supported canonical form, and enforcement of the associated comparison semantics, are performed by implementations that support this extension. Pyang does not enforce “mac-48” but implementation code can.
- Allows for other canonical formats mac-64, ip-32, ip-128 etc.
- Works with all MAC address definitions.
- Does need YANG support for seamless processing, of keys, lists and comparison.
- We could ask for YANG Next to enable checking identities in extensions.

```
leaf address {
    type ieee:mac-address;
    yang:canonical-form "yang:mac-48";
    mandatory true;
    description
     "A sample IEEE MAC address.";
  }
```

Implementation for IETF – not part of this

The canonicalization procedure which would have to be implemented in YANG is:

1. Validate the input against the lexical pattern.
2. Remove all separators.
3. Convert hexadecimal digits to uppercase.
4. Use the resulting 12 hexadecimal digits as the canonical form of the MAC address.

Equivalent inputs include:

```
aa:bb:cc:dd:ee:ff
AA:BB:CC:DD:EE:FF
aa-bb-cc-dd-ee-ff
AA-BB-CC-DD-EE-FF
```

These values yield the same canonical form:

```
0xAABBCCDDEEFF
```

A step to a complete solution

- From a YANG syntax perspective, the solution is complete. However, YANG processing would need to be modified to read the extension behavior for :
  - List keys uniquness
  - Leaf-list uniquness
  - MAC address Comparison
  - X-path comparisons

Comments?

Backup: Alternative to an identity.

- Rather than defining an identity an algorithm registry could be setup.
- The extension becomes the algorithm.
- This option is still not validated by pyang it is parsed as a string and has the additional need of a registry.