YANG Type Pattern Proposals for P802.1ASdn/D2.2

Author: Johannes Specht
Affiliation: Self
Date: July 16, 2024

About this Document

This document is an individual contribution in support of the comment resolution of P802.1ASdn/D2.2. It contains proposals for replacing the YANG patterns of YANG data type definitions scaled-ns, uscaled-ns and float64, and adjusting the associated YANG descriptions and references accordingly.

The proposals goe back to the rogue comment on the right column on page 5 of https://ieee802.org/1/files/private/asdn-drafts/d2/802-1ASdn-d2-0-dis-v01.pdf, which was rejected due to its unspecific nature.

This document is very specific in the sense that it provides YANG code that can be used as copy&paste replacements. Additional notes and explanations are provided for each replacement, providing additional background information and remarks to the ballot resolution group.

Contents

Specific Proposals

scaled-ns Type Definition

Current YANG Code

```
typedef scaled-ns {
  type string {
    pattern "[0-9A-F]{2}(-[0-9A-F]{2}){11}";
  }
  description
    "The IEEE Std 802.1AS ScaledNs type represents signed values of
    time and time interval in units of 2^16 ns, as a signed 96-bit
    integer. Each of the 12 octets is represented as a pair of
    hexadecimal characters, using uppercase for a letter. Octets are
    separated by a dash character. The most significant octet is first.";
  reference
    "6.4.3.1 of IEEE Std 802.1AS";
}
```

Proposed new YANG Code

```
typedef scaled-ns {
  type string {
    pattern " 0x[0-9A-F]{4}( [0-9A-F]{4}){5}";
  }
  description
    "The IEEE Std 802.1AS ScaledNs type represents signed values of
    time and time interval in units of 2^16 ns, as a signed 96-bit
    integer. The canonical and lexical representations are as
    specified in 6.4.3.1 of IEEE Std 802.1AS (i.e., five upper case
    hexadecimal words with 4 digits each and the words separated by
    single whitespace characters.”;
  reference
    "6.4.3.1 of IEEE Std 802.1AS";
}
```

Notes and Explanations

The original proposal from the rogue comment described two different patterns:

- A first pattern aligned with the notation illustrated in 6.4.3.1 of IEEE Std 802.1AS
- A second pattern aligned with the notation of Integers in YANG (9.2.1. of RFC 7950).

The proposed new YANG code limits on the former pattern as a result of discussion with 802.1AS experts.

uscaled-ns Type Definition

Current YANG Code

```
typedef uscaled-ns {
  type string {
    pattern "[0-9A-F]{2}(-[0-9A-F]{2}){11}";
  }
  description
    "The IEEE Std 802.1AS UScaledNs type represents unsigned values of
    time and time interval in units of 2^16 ns, as an unsigned 96-bit
    integer. Each of the 12 octets is represented as a pair of
    hexadecimal characters, using uppercase for a letter. Octets are
    separated by a dash character. The most significant octet is first";
  reference
    "6.4.3.2 of IEEE Std 802.1AS";
}
```

Proposed new YANG Code

```
typedef uscaled-ns {
  type string {
    pattern " 0x[0-9A-F]{4}( [0-9A-F]{4}){5}";
  }
  description
    "The IEEE Std 802.1AS UScaledNs type represents unsigned values of
    time and time interval in units of 2^16 ns, as an unsigned 96-bit
    integer. The canonical and lexical representations are as
    specified in 6.4.3.2 of IEEE Std 802.1AS (i.e., five upper case
    hexadecimal words with 4 digits each and the words separated by
    single whitespace characters.";
  reference
    "6.4.3.2 of IEEE Std 802.1AS";
}
```

Notes and Explanations

The notes and explanations for scaled-ns apply equally for uscaled-ns.

float64 Type Definition

Current YANG Code

```
typedef float64 {
 type string {
   pattern "[0-9A-F]{2}(-[0-9A-F]{2}){7}";
 }
 description
    "The IEEE Std 802.1AS Float64 type represents IEEE Std 754 binary64. Each of the 8 octets is
    represented as a pair of hexadecimal characters, using uppercase for a letter. Octets are
    separated by a dash character. The most significant octet is first.";
  reference
    "6.4.2 of IEEE Std 802.1AS";
}
```

Proposed new YANG Code

```
typedef float64 {
  type string {
    pattern "([+-]?0[Xx]([0-9a-fA-F]*.[0-9a-fA-F]+|[0-9a-fA-F]+.|[0-9a-fA-F]+)[Pp][+-]?[0-9]+)"+
            "|([+-]?([0-9]*.[0-9]+|[0-9]+.|[0-9]+)[Ee][+-]?[0-9]+)";
  }
  description
    "The IEEE Std 802.1AS Float64 type represents IEEE Std 754 binary64. The lexical
    representation is either that of external hexadecimal-significand character sequences
    representing finite numbers as specified in 5.12.3 of IEEE Std 754-2019, or that of
    ISO/IEC 9899:1999 (C99) for decimal floating-point numbers with exponent, without
    floating-suffix and limited to finite representable numbers (e.g., Inf. and NaN excluded).

    Canonical form:
     a)  The canonical form of a positive number does not include the sign '+', and does not
         include the sign '+' for positive exponents.
     b) The hexadecimal/decimal point is required.
     c) Lexically representable numbers that cannot be represented by binary64 shall be rounded
        by a server to the closest finite number representable by binary64.
     d) When a server sends XML-encoded data, only normalized values and are sent in the format
        according to 5.12.3 of IEEE Std 754-2019 with at least one fractional digit and one
        exponent digit (i.e., pattern '-?0[Xx]1.[0-9a-fA-F]+[pP]-?[0-9]+').
     e) XPath expression evaluations are done using the canonical form specified in items a)
        through d).";
  reference
    "6.4.2 of IEEE Std 802.1AS
```

```
    5.12.3 of IEEE Std 754-2019
    6.4.4.2 of ISO/IEC 9899:1999";
}
```

Notes and Explanations

The current YANG code is effectively a byte-wise memory dump of binary64 values.

This proposed new YANG code is intended to address several issues with this. This proposal is intended to address various usability concerns of memory-dumps that were raised in other contexts in IEEE 802.1, and likewise addresses potential issues with non-representable/non-finite numbers (e.g., NaN).

It is **to be discussed** whether a hexadecimal representation would be sufficient (i.e., omitting decimal entirely, which means all red text would disappear).

The proposed new YANG code is more aligned with a human readable form. A similar approach is found in "ietf-routing-types.yang" (https://www.netconfcentral.org/modules/ietf-routing-types/2017-12-04) for data type "bandwidth-ieee-float64". "bandwidth-ieee-float64" limits to hexadecimal, non-negative, normalized, non-fraction numbers. These limits are not implemented in "float64". Limiting to representable numbers is present in both, "float64" and "bandwidth-ieee-float64".

The proposed pattern does not account for potential values that (even) exceed the range of binary64. However, this should be covered by the definition of the canonical form. A simple (but naive) way for covering this in the pattern would be by liming the number of digits. However, this would only narrow the range limits. Accurate range limitation by pattern would be possible (in theory) by excessive combinatorial expansion of digit-combinations. This was omitted in favor of readability, and because the range limits should be covered, as said.

Notes on the canonical form:

- The canonical form requirements a) and b) are derived from 9.3.2. of IETF RFC 7950. Note that RFC 7950 does not limit to either upper- or lower- case letters for hexadecimal integer values (9.2.1 and 9.2.2 of RFC 7950).
- Item c) covers various cases for config data.
- Items d) and e) are derived from 9.1. of RFC 7950, though the normalization is a logical consequence to ease XPath evaluation.