Maintenance Request 0366: IEEE 802.1Qcw REVISION REQUEST on allowing non-scheduled interfaces

Florian Kauer (Linutronix) Tobias Deiminger (Linutronix)

June 18, 2024

Maintenance Request 0366: IEEE 802.1Qcw REVISION REQUEST on allowing non-scheduled interfaces

Several mistakes in the must statements of IEEE 802.1Qcw YANG models prevent their use in real-world scenarios.

In particular, a trivial interface like

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "if-index": 1,
        "name": "eth0",
        "type": "iana-if-type:ethernetCsmacd",
        "oper-status": "down",
        "admin-status": "down",
        "statistics": {
          "discontinuity-time": "2023-12-15T10:04:12.345+00:00"
        }
      }
    ]
  }
}
```

no longer validates when just loading the IEEE 802.1Qcw YANG models.

See https://www.802-1.org/items/473

One of the problematic cases in ieee802-dot1q-sched.yang

```
...
  leaf supported-list-max {
    type uint32;
    ...
  }
  ...
  container admin-control-list {
    must
      "(count(./gate-control-entry) <= ../supported-list-max)" {
      error-message
        "Number of elements in admin-control-list must not be greater"+
        "than supported-list-max";
    }
    ...
  }
...
```

The `must` statement is always evaluated even if no `admin-control-list` exists and always fails to validate if `../supported-list-max` is missing!

The presence statement

Why is it irrelevant if the `admin-control-list` exists?

In ‘7.5.5. The “presence” statement’ of RFC7950:

```
If a container has the "presence" statement, the container's
existence in the data tree carries some meaning.
```

→ If a container is a non-presence container, the container’s existence in the data tree carries no meaning.

→ The parser needs to handle a non-presence container in the same way regardless of its existence in the data tree.

The must statement

Why is the `must` statement always evaluated?

In '7.5.3. The "must" Statement' of RFC7950:

```
When a datastore is validated, all "must" constraints are
conceptually evaluated once for each node in the accessible tree.
(see Section 6.4.1).
```

In '6.4.1. XPath Context' of RFC7950:

```
If a node that exists in the accessible tree has a non-presence
container as a child, then the non-presence container also exists in
the accessible tree.
```

→ Non-presence containers are nodes in the accessible tree.

→ Must statements of non-presence containers must be evaluated, even if they are not in the datastore!

Evaluation of must statements

What is the result of `count(./gate-control-entry) <= ../supported-list-max` if no `../supported-list-max` exists?

In ‘7.5.3. The “must” Statement’ of RFC7950:

```
The "must" statement, which is optional, takes as an argument a
string that contains an XPath expression (see Section 6.4).
```

In ‘6.4. XPath Evaluations’ of RFC7950:

```
YANG relies on XML Path Language (XPath) 1.0 [XPATH] as a notation
for specifying many inter-node references and dependencies.
```

In ‘3.4 Booleans’ of XML Path Language (XPath) Version 1.0:

```
If one object to be compared is a node-set and the other is a number, then the comparison
will be true if and only if there is a node in the node-set such that the result of performing
the comparison on the number to be compared and on the result of converting the string-value
of that node to a number using the number function is true
```

→ If the node-set `../supported-list-max` is empty (i.e. not provided in the datastore), there is no chance for `(count(./gate-control-entry) <= ../supported-list-max)` to be true.

→ As soon as this must statement is somewhere in the accessible tree but `../supported-list-max` is not provided, any instance data is invalid!

Possible Solutions

- Invert statements: `not(count(./gate-control-entry) > ../supported-list-max)` (proposed in maintenance request)
- Make everything presence containers, then the must statements are only applied if the container exists.
- Provide default values to all relevant leafs.
- Add zeros `(count(./gate-control-entry) <= ../supported-list-max + 0)` (not proposed, but interesting corner case)

Proposed Solution

`not(count(./gate-control-entry) > ../supported-list-max)`

- Mathematically equivalent to the original for the case that `../supported-list-max` exist.
- If `../supported-list-max` is missing, the statement inside the `not(...)` is `false` regardless of the operator used, so `not(false) = true` .

→ No change in behavior for instance data with correct `sched` data, but the trivial example now also validates.

Adding Zeros (not proposed)

```
(count(./gate-control-entry) <= ../supported-list-max + 0)
```

Since we have a numeric operator on the right side now:

In '3.5 Numbers' of XML Path Language (XPath) Version 1.0:

```
The numeric operators convert their operands to numbers as if by calling the number function.
```

In '4.4 Number functions' of XML Path Language (XPath) Version 1.0:

```
a node-set is first converted to a string as if by a call to the string
function and then converted in the same way as a string argument
```

In '4.2 String functions' of XML Path Language (XPath) Version 1.0:

```
A node-set is converted to a string by returning the string-value of the node in the node-set
that is first in document order. If the node-set is empty, an empty string is returned.
```

In '4.4 Number functions' of XML Path Language (XPath) Version 1.0:

```
a string that consists of optional whitespace followed by an optional minus sign followed
by a Number followed by whitespace is converted to the IEEE 754 number that is nearest
(according to the IEEE 754 round-to-nearest rule) to the mathematical value represented
by the string; any other string is converted to NaN
```

$\rightarrow$ Unclear if empty string is 0 or NaN, but libc's strtold returns 0 without error and thus also libyang. With 0, $0 <= 0 + 0$, so true. Works with libyang, but depends on implementation...

Other Occurences

```
       container oper-control-list {
         must
-          "(count(./gate-control-entry) <= ../supported-list-max)" {
+          "not(count(./gate-control-entry) > ../supported-list-max)" {
           error-message
             "Number of elements in oper-control-list must not be greater"+
+            "Number of elements in oper-control-list must not be greater "+
             "than supported-list-max";
         }
```

In sched and psfp, equivalent to first example, just for oper-control-list.

In `ieee802-dot1q-psfp.yang`

```
...
list stream-gate-instance-table {
...
   refine "gate-control-entry/time-interval-value" {
     must
-          "(. <= ../../../supported-interval-max )";
+          "not (. > ../../../supported-interval-max )";
   }
...
}
...
```

- Very similar, but since the must is in a descendant of a list, it does not trigger for a trivial case, but just as soon as an element exists in the list.
- Same statement in sched, but without the list.

```
container admin-cycle-time {
  must
    "(./numerator div ./denominator <= "+
    "../supported-cycle-max/numerator div "+
    "../supported-cycle-max/denominator )" {
    error-message
```

- In sched and slightly different in psfp. The operation makes numbers out of both sides, so the reasoning of the first example does not apply!
- If all leafs do not exist and thus are parsed as 0 or NaN (depending on interpretation, see above), the result should be NaN <= NaN which should be false! The invesion with not() gives true, so that is OK.
- Note that 0/0 (which would wrongly be considered as valid as well) is not possible when the leafs do exist due to the allowed range for the denominator (1..4294967295).

```
        list flow-meter-instance-table {
          must
-           "(count(.) <= ../max-flow-meter-instances)" {
+           "not (count(.) > ../max-flow-meter-instances)" {
            error-message
-             "Number of elements in flow-meter-instance-table must not be"+
+             "Number of elements in flow-meter-instance-table must not be "+
              "greater than max-flow-meter-instances.";
          }
```

- In psfp. Similar to first example.
- However, the must statement is under the list which makes it part of each list entry and not of the list as a whole! Therefore, it needs to be moved to the enclosing container!

Validation

https://github.com/Linutronix/yang-validation

- Contains testcases that trigger all the mentioned problems as well as sched and psfp examples that work before and after the proposed change
- Testcases are automatically executed (Github Actions) when adding/changing testcases
- Also executed daily to be notified early about changes in https://github.com/YangModels/yang
- Maybe also useful for other scenarios $\rightarrow$ Open to ideas & contributions!