YANG 1.1

draft-ietf-netmod-rfc6020bis-08

IETF 94
Martin Björklund
mbj@tail-f.com

YANG 1.1 status

WG collected 60 issues at:

https://svn.tools.ietf.org/svn/wg/netmod/yang-1.1/issues.html

All issues either marked as dead or review. No open issues.

WGLC was for 07, three in-depth reviews, most comments addressed in -08.

Some open issues reported on new functionality (need to fix them), some on old functionality (not sure if we should discuss them) and some suggestions for new functionality (what do we do with these?)

Summary of issues

- *New function:* if-feature and default
- *New function:* accessible tree in when evaluation
- *Old function:* augment mandatory nodes
- *Old function:* unique module names
- *New feature:* non-unique leaf-list in config false
- *New feature:* key-less lists and non-unique leaf-lists in config true
- *New feature:* change semantics of the *choice* and *when* statements

New function: if-feature and default

```
  leaf foo {
    type enumeration {
      enum blue { if-feature blue; }
      enum white;
    }
    default blue;
 }
```

A. Make this illegal.

B. Allow if-feature in default.

C. Legal, but it means a server that support the leaf MUST support the feature.

D. (implicit variant of B) Legal – it means that the default value is used only if the feature is advertised.

New function: accessible tree in when

```
augment /… {
  when “foo = 42”;
  leaf foo { … }
}
```

The problem is that the when expression makes “foo” conditional, based on the value of “foo”.

Proposed solution: tentatively remove the conditional nodes while evaluating the when expression.

Concern raised: this might make it hard to understand what's going on

Old function: augment mandatory nodes

This is issue Y26 in the issues list. WG consensus was to keep current rule – it is illegal to augment mandatory nodes. This rule exists in order to protect clients that do not know the augmenting module.

New proposal: allow augment of mandatory nodes only in combination with a “when” condition, and only if the “when” condition is “safe” for the client.

Concern: “safe” for the client cannot be formally checked by a compiler

Old function: unique module names

Current text says:

> The names of all standard modules and submodules MUST be unique.
> Developers of enterprise modules are RECOMMENDED to choose names for their modules that will have a low probability of colliding with standard or other enterprise modules, e.g., by using the enterprise or organization name as a prefix for the module name.

Issue raised: isn't it the case that all module names MUST be unique?

Reality: all module names MUST be unique within a server

Compromise Proposal on the ML:

> Use of enterprise modules with non-unique names is NOT RECOMMENDED.

Old function: augment mandatory nodes

This is issue Y26 in the issues list. WG consensus was to keep current rule – it is illegal to augment mandatory nodes. This rule exists in order to protect clients that do not know the augmenting module.

New proposal: allow augment of mandatory nodes only in combination with a “when” condition, and only if the “when” condition is “safe” for the client.

Concern: “safe” for the client cannot be formally checked by a compiler

New feature: non-unique config false leaf-lists

YANG allows config false lists w/o keys:

```
list sample {
  leaf value { … }
}
<sample><value>10</value></sample>
<sample><value>20</value></sample>
<sample><value>10</value></sample>
```

However, config false leaf-lists must contain unique values.

Proposal: Allow non-unique leaf-lists in config false. Requires a new keyword.

New feature: keyless lists and non-unique leaf-lists in config

Proposal: Allow keyless lists and non-unique leaf-lists in config. When editing such list it can only be changed in its entirety. Individual list entries cannot be changed separately.

Comment: Similar to issue Y57 (non-unique leaf lists) which was discussed at length, and the WG decided not to do.

Also, at this time there is not a concrete proposal available, e.g. it is not clear how this would actually work in the XML encoding.

New feature: change semantics of the choice and when statements

Proposal: Remove the auto-delete feature of choice and when. i.e., it would be the client's responsibility to make sure that when a case branch is created, the old one (if any) is deleted.

Comments:

- The proposal doesn't solve any known problem in current deployments.
- Huge impact on existing clients and servers.
- Unclear what the difference between *when* and *must* would be.