Embedded YANG Schemas

Ladislav Lhotka
⟨lhotka@nic.cz⟩

21 March 2018

Objective

Augment YLbis so that it is possible to embed a schema at an arbitrary container or list node of another (parent) schema.

Characteristics of the Proposed Solution

The solution is based on the **use-schema** mechanism described in *draft-ietf-netmod-schema-mount-08* but independent of the approach described in *schema-mount-09*.

Many aspects are intentionally similar to YANG augments. This means that the concept should be easier to understand, and tools can reuse most of the code that implements augments.

In particular:

- the target node in the parent schema (root of an embedded schema) is identified using a schema node identifier that is included in the specification of the embedded schema
- no YANG extension is needed
- the overall schema is described in a compact form of (augmented) YANG library data, i.e. suitable for saving in a file (see *draft-lengyel-netmod-yang-instance-data*), including in an RFC etc.

Augmented YLbis Schema

```
+--ro schema* [name]
|  +--ro name                    string
|  +--ro module-set*             -> ../../module-set/name
|  +--ro eys:embedded-schemas
|     +--ro eys:namespace* [prefix]
|     |  +--ro eys:prefix    yang:yang-identifier
|     |  +--ro eys:uri?      inet:uri
|     +--ro eys:embedded-schema* [target]
|        +--ro eys:target              schema-node-id
|        +--ro eys:use-schema          -> /yanglib:yang-library/schema/name
|        +--ro eys:when?               yang:xpath1.0
|        +--ro eys:config?             boolean
|        +--ro eys:parent-reference*   yang:xpath1.0
```

- leafref **`eys:use-schema`** now points to a YLbis schema entry.
- schema embedding may be applied recursively

Embedded LNE Schema

```
"schema": [
  {
    "name": "lne-schema",
    "module-set": ["lne-modules"]
  },
  {
    "name": "physical-device-schema",
    "module-set": [ "physical-device-modules" ],
    "ietf-embedded-yang-schema:embedded-schemas": {
      "namespace": [
        {
          "prefix": "lne",
          "uri": "urn:ietf:params:xml:ns:yang:ietf-logical-network-element"
        }
      ],
      "embedded-schema": [
        {
          "target":
            "/lne:logical-network-elements/lne:logical-network-element/lne:root",
          "use-schema": "lne-schema"
        }
      ]
    }
  }
]
```

YANG Instance Data

Goal: validate YANG instance data snippets as described in *draft-lengyel-netmod-yang-instance-data* using standard YANG tools.

```
module ietf-yang-instance-data {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data";
  prefix yid;
  ...
  container instance-data {}
}
```

```
"schema": [
  {
    "name": "arbitrary-schema",
    "module-set": [ "any-modules" ]
  },
  {
    "name": "instance-data-schema",
    "module-set": ["instance-data-module"],
    "ietf-embedded-yang-schema:embedded-schemas": {
      "namespace": [
        {
          "prefix": "yid",
          "uri": "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data"
        }
      ],
      "embedded-schema": [
        {
          "target": "/yid:instance-data",
          "use-schema": "arbitrary-schema"
        }
      ]
    }
  }
]
```

SNI Versus YANG Extension

Embedded schemas use schema node identifiers for locating the target, schema mount uses the `mount-point` extension.

However:

- Augments also use SNIs; it doesn't seem necessary to formally indicate "augment points".
- The presence of the extension doesn't guarantee that something appears at that place – it depends on the implementor that composes YANG library data.
- It is not guaranteed that external stuff does **not** appear in other places (because we have augments).