Schema Mount Proposals

draft-bjorklund-netmod-structural-mount-02

draft-lhotka-netmod-ysdl-00

Martin Björklund
⟨mbj@tail-f.com⟩

Ladislav Lhotka
⟨lhotka@nic.cz⟩

4 April 2016
Why Schema Mount?

The existing YANG modularity/extensibility mechanisms don’t suffice for some common use cases.

```
+--rw interfaces
   |  +--rw interface* [name]
   |     |     ...
   |     |     +--rw ip:ipv4
   |     |     |     ...
   |     |     +--rw ip:ipv6
   |     |     ...
   +--rw logical-device* [name]
       +--rw name string
       |     ...
       +--rw interfaces
           +--rw interface* [name]
           |     ...
           +--rw ip:ipv4
           |     ...
           +--rw ip:ipv6
           ...
```

Neither grouping/uses nor augment represent a suitable solution.
Proposed Solutions

1. draft-bjorklund-netmod-structural-mount-02
2. draft-lhotka-netmod-ysdl-00

Conclusion of virtual interim meeting on 22 February 2016:

- combine these two proposals into a single consolidated solution,
- adopt an updated revision of #1 as a WG document.
## Terminology

**schema**  
defined in YANG modules specified by an instance of YANG library (including features and/or deviations).

**schema mount**  
results in a new schema constructed by inserting an existing schema in a specified location of a parent schema.

**mount point**  
a schema node in the parent schema under which the child schema is inserted.
Common Characteristics

- Both solutions satisfy the definition of schema mount, the server defines the mount point(s) and contents of a child schema(s) as separate read-only data.
- Each schema is isolated and self-contained, references to nodes, groupings, typedefs etc. that are defined outside a given schema aren’t permitted.
- The mount point, or the closest ancestor that is a data node, represents the conceptual root of the child schema data, and all absolute schema identifiers and XPath expressions are interpreted relative to it.
RPC Operations and Notifications

*structural-mount* proposes a generalisation of (module-level) RPC operations and notifications to non-root schemas: they are treated as actions and notifications connected to the corresponding mount point.

**Example:**

RPC operation *system-restart* is defined in *ietf-system* [RFC 7317].

```xml
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <action xmlns="urn:ietf:params:xml:ns:yang:1">
        <managed-devices xmlns="urn:example:network-manager">
            <device>
                <name>rtrA</name>
                <system xmlns="urn:ietf:params:xml:ns:yang:ietf-system">
                    <system-restart/>
                </system>
            </device>
        </managed-devices>
    </action>
</rpc>
```
Open Issues

1. specification of mount points,
2. relationship to YANG library instance(s),
3. permitted mount points,
specification of mount points

Options:

1. *structural-mount*: YANG extension *mount-point*.
   - Possible mount points can/must be specified by the parent module’s author.
   - A YANG extension is used for a fundamental construct.

2. *YSDL*: schema node identifier.
   - Similar to target node in an *augment*.
   - Requires no change to YANG modules.
relationship to YANG library

Options:

1. *structural-mount*: child schema specification refers to a YANG library instance mounted under the child schema mount point.
   - The client constructs the schema iteratively by looking up corresponding state data.
   - If the mount point is a *list*, different schemas may be mounted under different entries.
   - There is no concise description of the complete schema, it has to be constructed iteratively based on state data (acceptable for NMS, may be a problem for other tools).
2. **YSDL, structural-mount**: list of modules (+ features and deviations) is a part of child schema specification.

- The overall schema is still static and the client is able to determine it up front by reading a single instance of YANG library **and** specification of subschema structure.

- Schema variation for list entries can be achieved in YSDL by using **case** nodes in the parent schema as mount points. Possible extension: using **when**-like expressions as in augments.
permitted mount points

Options:

1. *structural-mount*: container, list.
2. *YSDL*: container, list, case, anydata.
3. Recent discussion in the mailing list: only *anydata*.
   - This makes the schema weaker: *anydata* as defined in 6020bis is not limited to data defined in the mounted schema.
   - Is it necessary? “Old clients” have to be prepared to handle unknown data in any *container* or *list* anyway that may be added through an augment from a module that the client doesn’t support.