“Populating a list of YANG data nodes using templates”

draft-rajaram-netmod-yang-templates.00

D. Rajaram (NOKIA), R. Peschi (presenting) (NOKIA), S. Ashraf (NOKIA)

IETF-121 NETMOD

Executive summary

General **template technique** to configure ”YANG managed devices”

- = typically hosting a NETCONF YANG Server
- For devices with many instances of same function with limited config. variations (e.g. device with same HW sub-system replicated many times)

Benefits:

- **Improves scalability**: smaller running Data Store -> less memory footprint & faster configuration
- **Reusability & consistency**: device applies a common structure multiple times
- **Simplifies maintenance**: template modification automatically reflected where used
- Relies on **existing YANG 1.1 and off-the-shelf tools**.

Draft as **“informational”** (no extension to NETCONF protocol nor YANG 1.1)

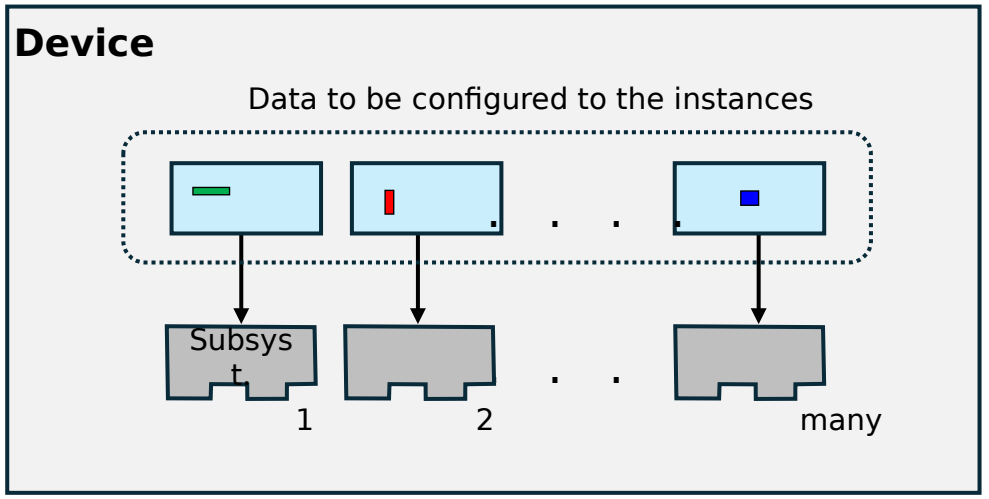
- -> context for other IETF documents that already use YANG template concepts for specific Use Cases, but however do not provide full context about the method.
  For instance draft-ietf-ccamp-optical-impairment-topology-yang and RFC 8795 Section-5.9.

Future draft: this template technique when **data nodes defined in existing IETF standards**

Scope of the draft

This draft describes a technique to configure a YANG device(*) such that

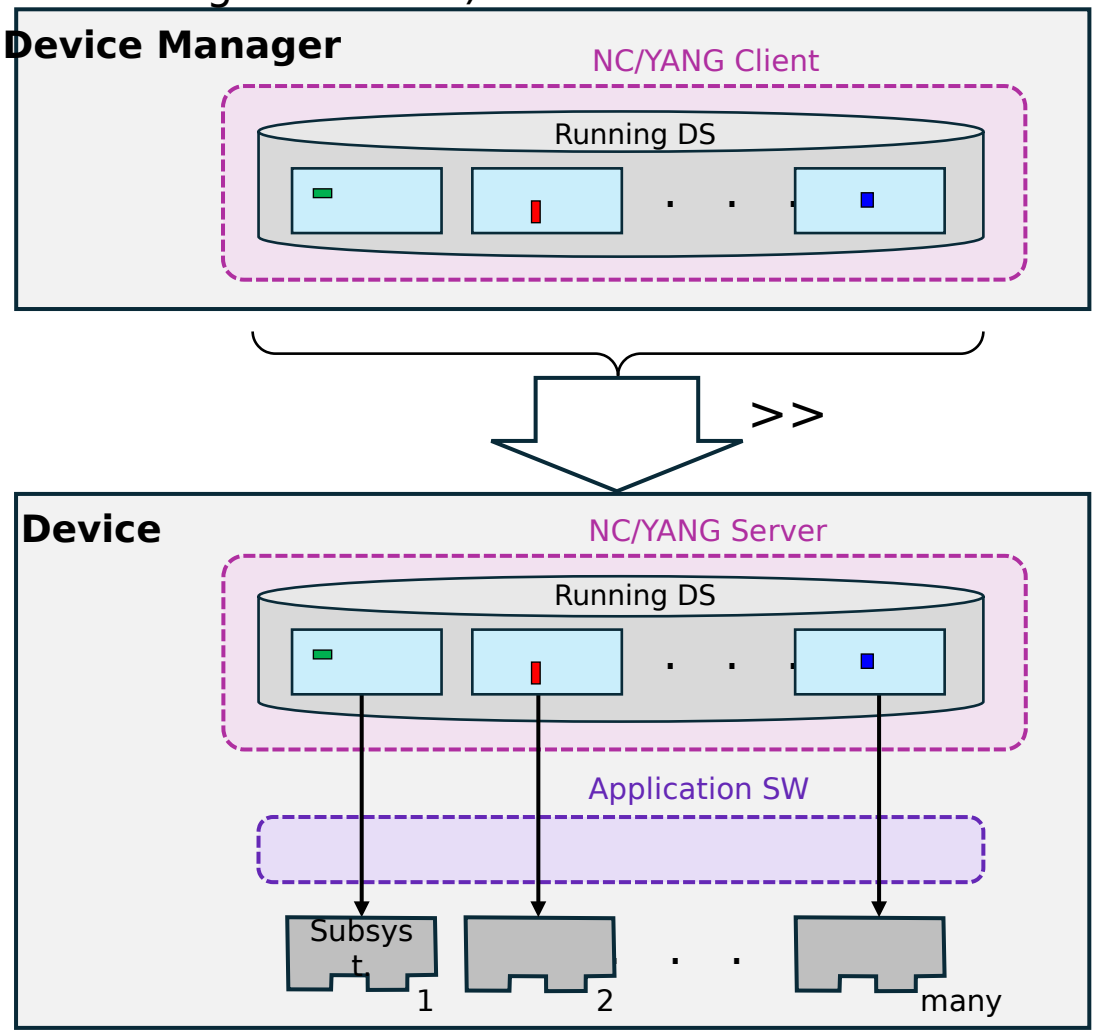
- has large list of identical ‘functional instances’, e.g. a HW subsystem replicated in the system.
- a functional instance has many data nodes to be configured
- only few data node value variations from instance to instance



(*) ”YANG device” = entity managed through a YANG model, i.e. typically hosting a *NETCONF YANG Server*

YANG configuration of the device: traditional way

- Device manager (e.g. NETCONF Client) configures device with each data node of each instance, one by one
- Inconvenient (error prone, long device configuration time, large device memory footprint due to large running data store)



Example of traditional YANG tree:

```
+--rw instance* [name]    // large list of functional instances
    +--rw name             string-ascii64
    +--rw description?     string-ascii128
    +--rw data
       +--rw list-a [name] //e.g. similar to a list of interf.
       |  +--rw name
       |  +--rw parm-x
       |  +--rw parm-y
       |
       +--rw list-b [name] //e.g. similar to a list of hw comp.
          +--rw name
          +--rw parm-t
          +--rw parm-u
```

Improving configuration process with a *YANG template* technique

Improves the traditional configuration of such device:

- reduces running data store size: less data to configure to device -> faster configuration time,
- helps consistency, less error prone
- relies on existing YANG 1.1 and available of-the shelf-tool

Other IETF documents already mention the use of YANG template concepts for specific applications, for instance:

- draft-ietf-ccamp-optical-impairment-topology-yang and
- https://datatracker.ietf.org/doc/html/rfc8795#section-5.9.

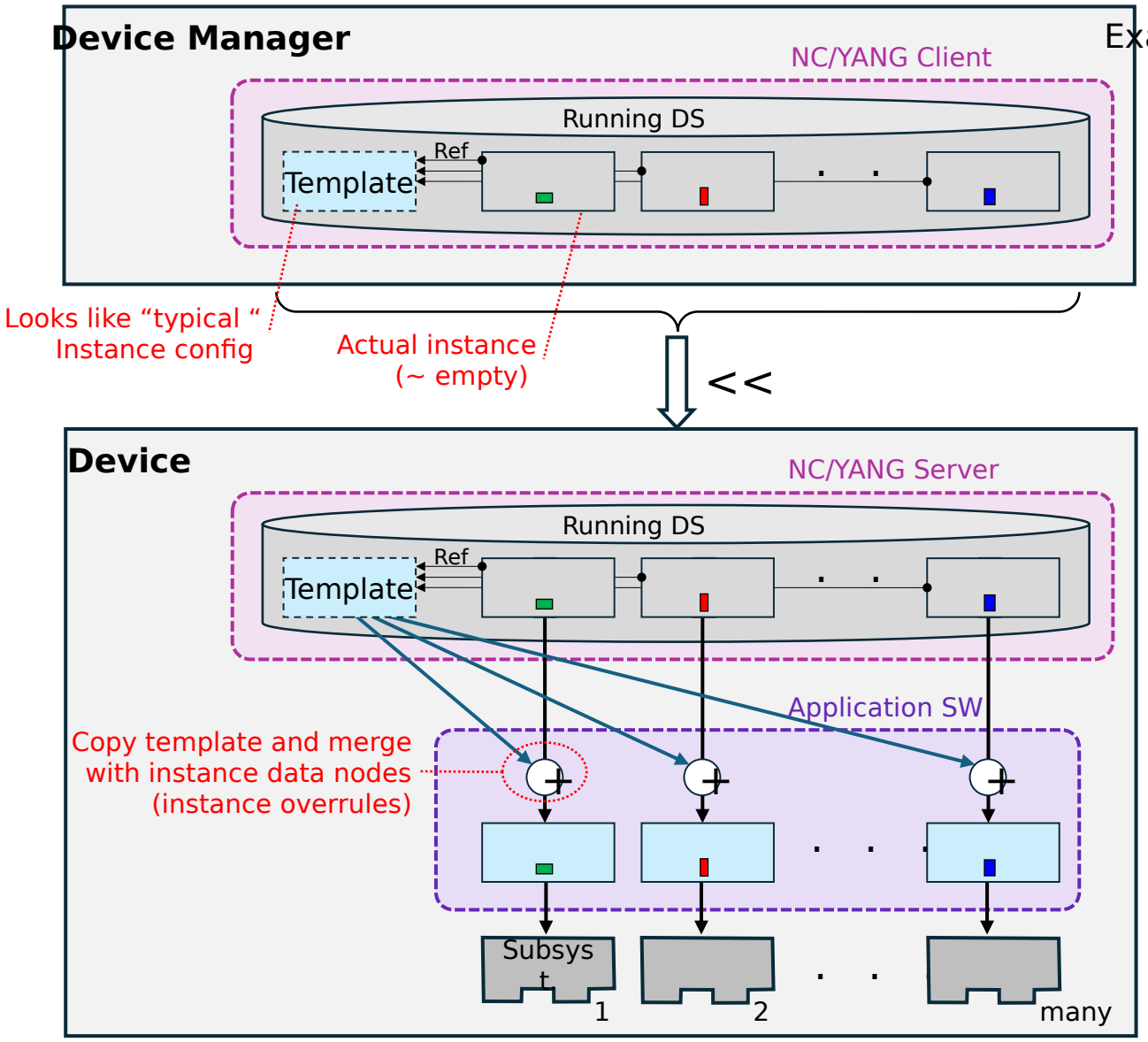
However, they don’t provide full context about the method.

This draft to clarify and formalize generic framework about how to define and use YANG templates.

Template technique: how it works

- configure a "***model" of a typical instance***, call it "***template***"
- configure list of instances with just a **reference** to a configured template. The reference instructs the device software to **copy** template configuration into the (empty) device configuration
- an instance may need **some variation** from the template configuration: the technique allows to configure any data nodes on a per instance basis with values that will **overrule** values coming from the template
  (NB: template method is most efficient when deviations from the template are limited).
- To allow **any** data node to be customized, **template** YANG model and **instance** YANG model contain the **same data nodes** (see further, though, about mandatory and default statements subtlety).
- A small **list of templates** can be defined to allow more drastic variations between group of instances

Configuring the device by means of template technique:



Example of YANG tree using the template technique:

```
+--rw template* [name]     // small list of templates
|  +--rw name              string-ascii64
|  +--rw description?      string-ascii128
|  +--rw data
|     +--rw list-a [name] //e.g. similar to a list of interf.
|     |  +--rw name
|     |  +--rw parm-x
|     |  +--rw parm-y
|     |
|     +--rw list-b [name] //e.g. similar to a list of hw comp.
|        +--rw name
|        +--rw parm-t
|        +--rw parm-u
|
+--rw instance* [name]     // large list of functional instancss
   +--rw name              string-ascii64
   +--rw description?      string-ascii128
   +--rw template?         -> /data-nodes-pattern/template/name
   +--rw data
      +--rw list-a [name] //e.g. similar to a list of interf.
      |  +--rw name
      |  +--rw parm-x
      |  +--rw parm-y
      |
      +--rw list-b [name] //e.g. similar to a list of hw comp.
         +--rw name
         +--rw parm-t
         +--rw parm-u
```

Mandatory and default statements in the YANG model of templates and instances

Data nodes with **default** or **mandatory** statements can counteract the template technique if kept in the instance:

- **Defaults** in the instance will silently and unintentionally overrule a value explicitly configured in the template due to the merge operation.
- **Mandatory** data nodes must be unconditionally configured in the instance although they were already configured in the template, -> reduces efficiency of template mechanism.

-> same data nodes for templates and instances, **but instance data nodes must be without default and mandatory statements.**

Easy way to go is when data nodes are defined in *grouping(s) without any mandatory or default statement*.

- In the instance YANG, grouping(s) suitable for use as-is,
- In the template YANG, grouping(s) refined by adding mandatory and default statements where appropriate.

Summary and next step

General **template technique** to configure YANG managed devices

- For devices with many instances of same function with limited config. variations (e.g. device with same HW sub-system replicated many times)

Benefits:

- **Improves scalability** : smaller running Data Store -> less memory footprint & faster configuration
- **Reusability & consistency**: device applies a common structure multiple times
- **Simplifies maintenance**: template modification automatically reflected where used
- Relies on **existing YANG 1.1 and off-the-shelf tools**.

Draft **as “informational”** (cf “YANG application note”)

- -> context for other IETF documents that already use YANG template concept for specific Use Cases, but however do not provide full context about the method.
  For instance draft-ietf-ccamp-optical-impairment-topology-yang and RFC 8795 Section-5.9.

Future draft: this template technique when **data nodes defined in existing IETF standards**