IETF NETMOD Virtual Interim

System-defined Configuration

draft-ietf-netmod-system-config-04

January 23, 2024

Recap/Overview

Draft-ietf-netmod-system-config defines how a management client and server handle YANG-modeled config data that is defined by the server itself.

- A “system” datastore (config true, read-only) to hold system config
- Clients may reference, override system config and configure descendant nodes of system config
- <system> flows into <intended> with <running> taking precedence over it
  - after config transformations defined in RFC 8342
- Two ways to satisfy referential integrity constraints in <running>
  1. Explicit declaration of referenced system nodes
  2. Using the “resolve-system” parameter

Two issues that this presentation is about

1. The “origin” issue

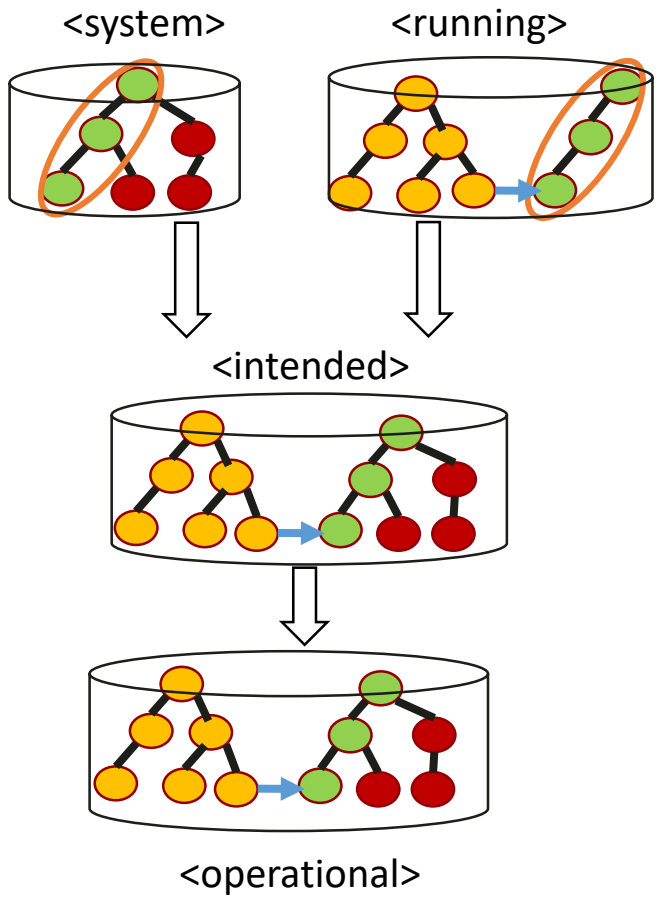
2. Validity of <running> alone

The “origin” issue for system nodes copied into <running>

This issue is independent of the discussion about validity of <running> alone

reference, e.g., leafref

nodes copied from <system> into <running>



“origin” reported as “system”

“origin” reported as “intended”

“origin” reported as ???

The “origin” issue for system nodes copied into <running> (cont.)

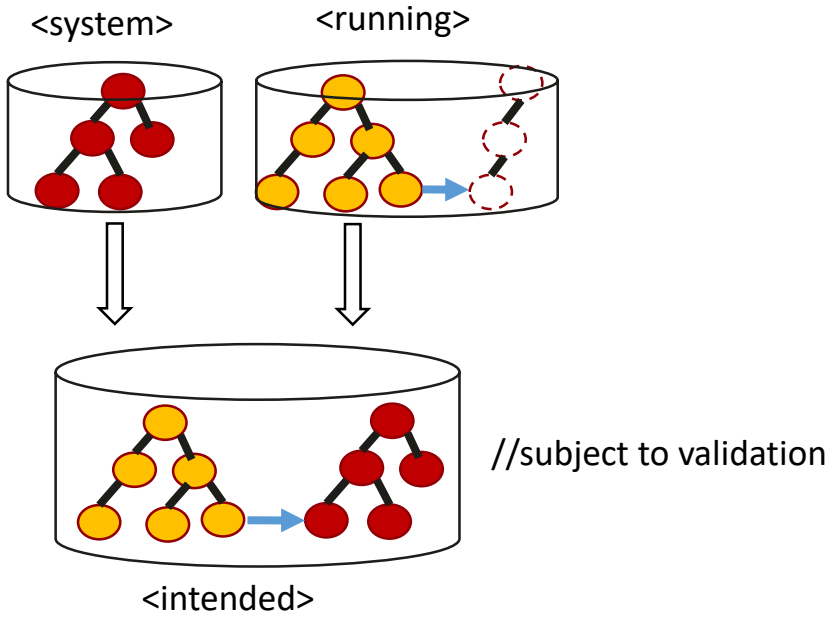
- The question behind this issue is whether we want a copied system node in <running> to always override and take precedence over <system>
  - What if the copied system nodes are immutable (e.g., an interface type)? This is the case where config in <running> should not take precedence.

- Servers’ upgrade migration might only happen on nodes with certain origin value.

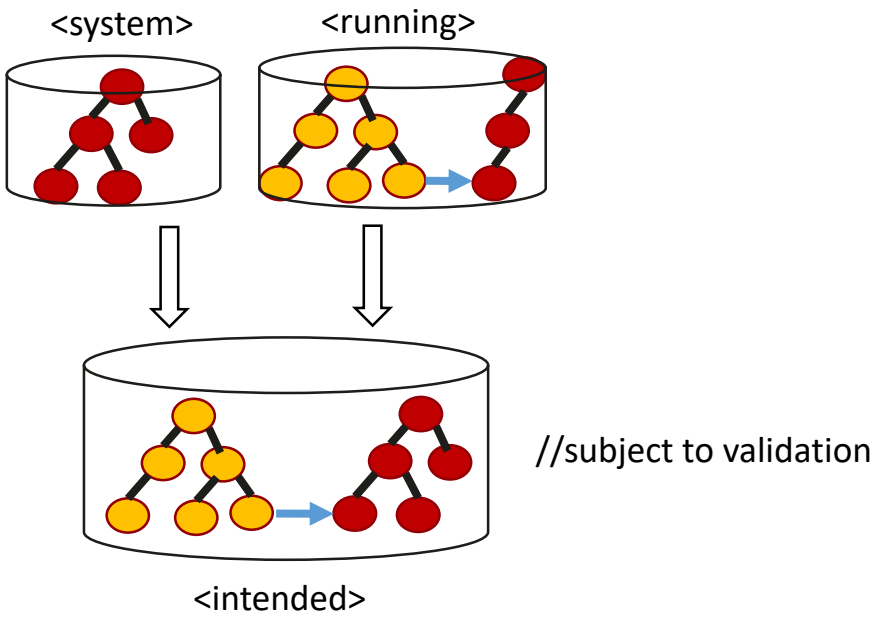
The Key Issue: validity of <running> alone

The question is whether we always need a referentially complete configuration in <running>.

- reference, e.g., leafref



Option 1
References not copied into <running>



Option 2
References copied into <running>

Online validation vs. Offline validation

Not defined in any of existing RFCs, but both are heavily mentioned in previous discussions and in this presentation

- Online validation:

  The protocol operation (i.e., <validate>) enforced at the server side to validate the contents of the specified configuration

- Offline validation:

  The operation performed by the client/offline tools to validate the configuration fetched from the server

Some Quotations from existing RFCs

Sec.8.1. Constraints on Data (RFC 7950)

"The running configuration datastore MUST always be valid."

Sec.5.1.3. The Running Configuration Datastore (<running>) (RFC 8342)

"However, <running> MUST always be a valid configuration data tree, as defined in Section 8.1 of [RFC7950]."

Sec.5. Architectural Model of Datastores (RFC 8342, Figure 2)

<intended> // subject to validation

Sec.5.1.4. The Intended Configuration Datastore (<intended>) (RFC 8342)

"…, but <intended> MUST always be a valid configuration data tree, as defined in Section 8.1 of [RFC7950]."

Long-term consideration of validity of <running> alone

Moving to NMDA's interpretation of Validity of <running> Alone…

- "Long-term" means YANG-next/NETCONF-next/RESTCONF-next
- NMDA's interpretation from a long-term's perspective
  - Validation is on <intended> (because <running> alone is incomplete)
  - Online validation: <running> is valid implicitly if <intended> is valid
  - Offline validation of <running> necessitates clients:
    1. Perform the configuration transformation to <running>
       - e.g., template expansion, inactive config removal
    2. Merge <running> into <system> to become <intended> and validate resulting <intended>
       - for <system> aware clients: easy to achieve
       - for <system> unaware clients: needs proprietary mechanism to fetch system configuration
- Assume RFC 8342 (NMDA) only applies to NMDA servers?
  - Assume clients connecting to NMDA-server know to validate <intended>?

Can we do this now?

i.e., not wait for YANG-next/NETCONF-next/RESTCONF-next

Options

1. Don’t require <running> alone is offline valid
2. Require <running> alone MUST be offline valid
3. Just state <running> MUST always be valid

Option 1

Don’t require <running> alone is offline valid

- Treat it as a clarification/bug-fix in existing RFCs
  - Might break legacy clients and tool chains, e.g., a client makes an “implicit” reference to system configuration, and legacy clients need to do offline validation of <running>
- Is it a prerequisite to first standardize the config transformation between <running> and <intended> ?
  - e.g., template expansion, inactive configuration removal

Option 2

Require <running> alone MUST be offline valid

- Are the complexities necessary? even if for system config, only the parts that are required to make <running> valid need to be copied (e.g., the key node)
- The server might need to migrate some nodes in <running> during the device upgrade if the config in <system> updates/disappears and a stale copy is in <running>
  - May be connected to the issue of “origin”
- How to handle config templates and inactive config if we require <running> alone MUST be offline valid?
  - E.g., inactive config is removed before validation?
- No non-backward compatible issues seen

Option 3: a possible compromise?

- Just state that <running> MUST always be a valid configuration data tree, as per RFC 8342 and RFC 7950

- That is, do not state explicitly referenced system configuration MUST always be copied into <running> or not
  - This leaves it up to interpretation (same as existing specifications)

The outcome of this discussion

- Having it explicitly stated would be good
  - Prevent future confusion
- Should RFC 8342 be “updated”?
  - To clarify that:
    - —<running> alone may be incomplete
    - —Clients connecting to an NMDA-server SHOULD expect this
    - —Legacy clients may experience validation errors