Updates to NMDA datastore architecture draft

draft-ietf-netmod-revised-datastores-03

Rob Wilton (Cisco), on behalf of NMDA authors

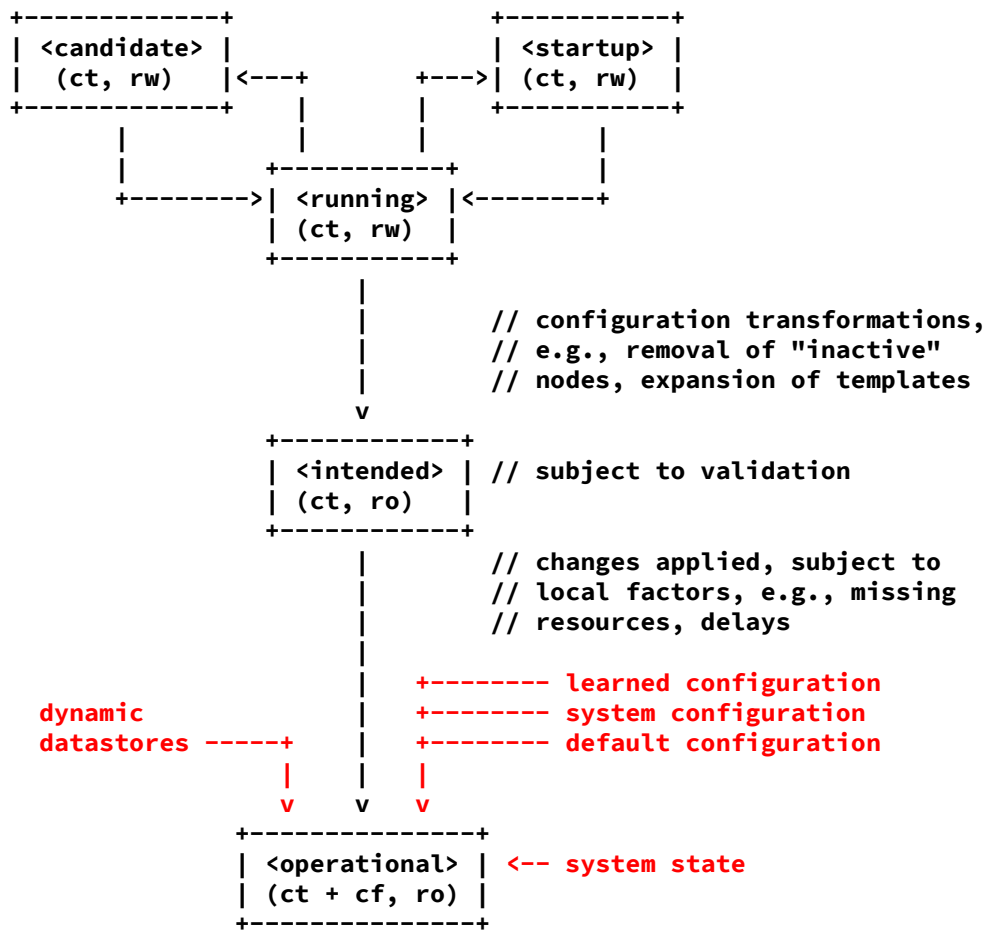
rwilton@cisco.com

IETF 99, Prague, Netmod WG

1 slide reminder of draft:

- Operator requirement for devices to clearly differentiate between:
  - What it is being **asked to do** – i.e. the **intended configuration**
  - What it is **actually doing** – i.e. **operational state, including the applied configuration**.
- Different solutions to this problem has been evaluated by IETF.
- The IETF solution defines a new datastore for operational state:
  - This has implications on the structure of YANG models to be simplified and optimized for use with NMDA.
  - Also replaces the existing ‘broken’ NETCONF GET operation.
  - NETCONF/RESTCONF additions to support the operational datastore.

Canonical datastores picture:



Summary of what has changed
(since -01, presented at Chicago)

1. Improved the definitions of configuration and state.
   Pulled in, and clarified, definitions of existing datastores currently defined in NETCONF RFC.
2. Further clarification on the semantics of <operational>.
3. Refinements of origin meta-data.
4. Clarified xpath usage in NMDA datastores.

(1) Refinement of definitions

- We have worked hard to improve the definitions of configuration and state
  - Defined “conventional configuration datastores”
  - Added learned configuration
  - Got rid of the “static configuration” term ᨒ
- Pulled in definitions for startup, candidate, and running datastores:
  - Aims to become the definitive reference for these.
  - Be more explicit on their exact semantics.
- *Please review these definitions carefully for correctness and completeness*

Refinement of definitions, “configuration”:

- Updated based on feedback on the list:

“*Data that is required to get a device from its initial default state into a desired operational state. This data is modelled in YANG using "config true" nodes. Configuration can originate from different sources.*”.

- Alternative way of thinking of this: “config: true means that the node **could** be configured via conventional datastores.

Refinement of definitions, “learned configuration”:

“*Configuration that has been learned via protocol interactions with other systems that is not conventional or dynamic configuration*”.

- E.g. if the operational state for a config true YANG node was acquired from BGP then that is “learned configuration”.
- Open issue:
  - Can the “learned” origin also apply to state nodes?

(2) Refined operational datastore definition:

- Clarified how defaults work in <operational>:

  “Requests to retrieve nodes from <operational> always return the value in use if the node exists, regardless of any default value specified in the YANG module. If no value is returned for a given node, then this implies that the node is not used by the device.”

- However, we probably still need to clarify what “*in use*” means:

  Trying to strike the right balance between being explicit (good) and not returning too much “noise” data (bad).

(3) Refinement of origin meta-data:

- Origin meta-data indicates **where** a data value has come from.
- Applies to all YANG nodes.
- Currently focus is on <operational>, but could be used in other datastores:
  - E.g. intended, or dynamic datastores
- Our intent is that it is optional to implement.

(3) Origin meta-data identities:

1. Intended – from intended datastore
2. Dynamic – from a dynamic datastore (except derived identities)
3. System – system configuration or system state (most prevelant for config false)
4. Learned – learned from a peer device
5. Default – default value from the schema.
6. Unknown – origin is unknown

(4) Refinement to xpath context

- The xpath expression for instance data in <operational> will resolve to <operational>.
- Xpath expressions for configuration in configuration datastores continues to resolve as before, i.e. the datastore the instance data resides in.
- Input/output parameters for notifications, RPCs, and action statements are evaluated in the context of <operational>.
  - The notifications, RPCs, actions could act on different datastores.

Open issues

1. Can “learned” origin apply to config false nodes?
2. Define “in use” nodes for <operational>
3. Should guidelines be in the body, and are they normative text?

- Issues above are quite minor, so ready for WG LC?