Refined YANG datastores Metadata
draft-wilton-netmod-refined-datastores-01
draft-wilton-netmod-opstate-metadata-00

IETF 96 – Berlin, NETCONF WG
Rob Wilton – Cisco
rwilton@cisco.com
Problem Description

Already presented in NETMOD:

• Two datastore drafts presented give a **conceptual vision** of how YANG datastores should evolve

• d.wilton-refined-datastores introduces **abstract datastores**

• d.wilton-opstate-metadata represents the abstract datastores using protocol agnostic YANG metadata.

• This presentation gives a quick recap, but focuses on the potential impact to NETCONF and RESTCONF.
Recap: draft-wilton Datastore Model

- **Candidate Configuration DS**
  - Writable via `<edit-config>`
  - Readable by `<get-config>`

- **Startup Configuration DS**

- **Persistent Configuration DS (r/w, ct)**

- **Ephemeral Configuration DS (r/w, ct)**

- **Applied Configuration abstract DS (ro, ct)**
  - System created config nodes (ct)
  - Config false nodes

- **Operational State Datastore (ro, ct + cf)**
  - Read/written by I2RS agent

- **Intended Configuration abstract Datastore (ro, ct + cf)**
  - Content returned in NETCONF `<get>` request
Recap of metadata draft

• The draft defines a YANG extension module that contains:
  o a **cfg-status** leaf to indicate the status of a configuration node
  o a **cfg-status-reason** leaf to indicate failure reasons

• Clients can request the metadata annotations are returned during get/get-config requests:
  o **Selectively-annotate** returns all nodes + metadata for all non converged nodes
  o **Annotate-all** return all nodes + metadata for all config nodes
  o **Annotated-diff** returns nodes + metadata for non converged nodes only

• Should also allow the same metadata to be made available during YANG pub/sub subscriptions
Proposed additions to NETCONF

- **2 new datastores:**
  - Persistent configuration
  - Operational state datastore (default target for `<get>`)  

- **3 new optional datastores:**
  - Ephemeral configuration
  - Intended configuration (no write operations)
  - Applied configuration (no write operations)

- **Support for opstate metadata:**
  - On persistent, ephemeral, operational state datastores

- **Support for running configuration datastore:**
  - `<edit-config>`, `<get-config>` map to “Persistent configuration DS”,
  - `<get>` maps to the “Operational state datastore”
Impact to opstate unaware NETCONF servers

- Defined as intended config == applied config
- `<get>` request now maps to the “operational state datastore” which may now include system controlled configuration nodes.
- Can support requests against the new datastores, meta-data, etc.
  - Mostly these operations should be handled as if the request was against the “running configuration datastore”.
Support for new servers via RESTCONF

• RESTCONF should allow datastore aware accesses:
  o Persistent configuration
  o Operational state datastore (default target for <get>)
  o Ephemeral configuration (optional)
  o Intended configuration (optional, no write operations)
  o Applied configuration (optional, no write operations)

• Support for opstate metadata:
  o On persistent, ephemeral, operational state datastores

• Existing RESTCONF request are mapped to new datastores:
  o PUSH/POST/etc go to the Persistent configuration datastore
  o GET requests go to the Operational state datastore
Questions? Comments?

- Thanks for listening.
- Any questions or comments?