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

IETF 96 – Berlin, NETMOD WG

Rob Wilton – Cisco

rwilton@cisco.com

Problem Description

- Two datastore drafts presented give a **conceptual vision** of how YANG datastores should evolve
- d.wilton-refined-datastores introduces **abstract datastores**
- Abstract datastores could be implemented as a regular client accessible datastore
- But this draft proposes a more efficient way of communicating the content of the abstract datastores via **YANG metadata annotations** to the:
    - **Persistent configuration datastore**
    - **Ephemeral configuration datastore**
    - **Operational state datastore**

Overview of solution draft

- The draft defines a YANG extension module that contains:
  - a **cfg-status** leaf to indicate the status of a **config true node**
  - a **cfg-status-reason** leaf to indicate failure reasons
- Clients can request the metadata annotations are returned during get/get-config requests:
  - **Selectively-annotate** returns all nodes + metadata for all non converged config nodes
  - **Annotate-all** return all nodes + metadata for all config nodes
  - **Annotated-diff** returns nodes + metadata for non converged config nodes only
- Should also allow the same metadata to be made available during YANG pub/sub subscriptions

cfg-status enum values

- The **cfg-status** leaf can take any of these values:
  - **Applying** – config node is in the process of being applied/deleted
  - **Applied** – config node has been applied
  - **Applied-deviation** – config node applied, but with deviant value
  - **Overridden** – config node overridden by another config datastore
  - **System-controlled** – config node exists only due to system
  - **Blocked** – config cannot be applied, hardware resource is missing
  - **Failed** – config failed, cfg-status-reason leaf indicates why.
- Not all enum values apply to all datastores
- **cfg-status-reason** leaf can provide extra information (for blocked or failed states)

Example <get> request against the **persistent configuration** datastore

```
<rpc message-id="101“
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <persistent/>
    </source>
    <filter type="subtree">
      <interfaces xmlns="http://example.com/ns/interfaces"/>
</filter>
    <with-applied-cfg-metadata>
     xmlns="urn:...:ietf-netconf-with-applied-cfg-metadata">
      selectively-annotate
    </with-applied-cfg-metadata>
  </get-config>
</rpc>
```

Example <get> response against the **persistent configuration** datastore

```
<rpc-reply message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:opstate="urn:...:ietf-yang-opstate-metadata">
  <data>
    <interfaces xmlns="http://example.com/ns/interfaces">
      <interface>
        <name>eth0/0</name>
        <mtu opstate:cfg-status="failed“
             opstate:cfg-status-reason="hardware programming error">
          9001
        </mtu>
      </interface>
      <interface>
        <name>eth0/1</name>
        <mtu opstate:cfg-status="applying">
          9000
        </mtu>
      </interface>
  </data>
</rpc-reply>
```

Example <get> request against the **operational state** datastore

```
<rpc message-id="102“
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <source>
      <opstate/>
    </source>
    <filter type="subtree">
      <interfaces xmlns="http://example.com/ns/interfaces"/>
    </filter>
    <with-applied-cfg-metadata>
     xmlns="urn:...:ietf-netconf-with-applied-cfg-metadata">
      selectively-annotate
    </with-applied-cfg-metadata>
  </get>
</rpc>
```

Get response against **operational state** ds

```
<rpc-reply … xmlns:opstate="urn:...:ietf-yang-opstate-metadata">
  <data>
    <interfaces xmlns="http://example.com/ns/interfaces">
      <interface>
        <name> eth0/0 </name>
        <mtu opstate:cfg-status="failed“
             opstate:cfg-status-reason="hardware programming error">
          1514
        </mtu>
        <enabled> true </enabled>
        <oper-status> up </oper-status>
      </interface>
      <interface>
        <name> eth0/1 </name>
        <mtu opstate:cfg-status="applying"> 1514 </mtu>
        <enabled> true </enabled>
        <oper-status> up </oper-status>
      </interface>
      <interface>
        <name opstate:cfg-status="system-controlled"> eth0/3 </name>
        <mtu opstate:cfg-status="system-controlled"> 1514 </mtu>
        <enabled opstate:cfg-status="system-controlled"> false </enabled>
        <oper-status> down </oper-status>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```

Questions? Comments?

- Thanks for listening.
- Further consideration of the impact of the refined datastores with metadata on NETCONF/RESTCONF will be considered in the NETCONF WG.
- Any questions or comments?
- Does the WG think that this is a good approach to modelling intended vs applied status?