NMDA Q & A

draft-dsdt-nmda-guidelines &
draft-ietf-netmod-revised-datastores-03

Rob Wilton (Cisco), on behalf of NMDA authors

rwilton@cisco.com

IETF 99, Prague, Netmod WG

Which RFCs need to be updated?

- RFC 6022: YANG Module for NETCONF Monitoring

  ietf-netconf-monitoring@2010-10-04.yang defines netconf-state

- RFC 7223: A YANG Data Model for Interface Management

  ietf-interfaces@2014-05-08.yang defines interface-state

- RFC 7277: A YANG Data Model for IP Management

  ietf-ip@2014-06-16.yang augments interface-state

- RFC 7317: A YANG Data Model for System Management

  ietf-system@2014-08-06.yang defines system-state

- RFC 7895: YANG Module Library

  ietf-yang-library@2016-06-21.yang defines module-state

…

Which RFCs need to be updated?

- RFC 8040: RESTCONF Protocol
  ietf-restconf-monitoring@2017-01-26.yang
  ietf-restconf@2017-01-26.yang
  defines restconf-state
- RFC 8022: A YANG Data Model for Routing Management
  ietf-ipv4-unicast-routing@2016-11-04.yang
  ietf-ipv6-router-advertisements@2016-11-04.yang
  ietf-ipv6-unicast-routing@2016-11-04.yang
  ietf-routing@2016-11-04.yang
  defines and augments routing-state
- Also: RFC 7758: Time Capability in NETCONF
  ietf-netconf-time@2016-01-26.yang augments netconf-state
  Experimental – won't immediately update.

Optional FAQ

Are the NETCONF/RESTCONF extensions backwards compatible?

Is NMDA only useful for large/complex routers?

What happens if I cannot align the config and state schema node?

When might intended and operational values deviate?

What if the “actual” value doesn’t conform to the schema constraints?

How does the data from dynamic datastores merge with into <operational>

Are the NETCONF/RESTCONF extensions backwards compatible?

- Yes, the plan is for minimal extensions to support NMDA.
- Details to be covered in NETCONF WG ᄊ
- V2 revisions of the protocols could be defined in future (probably with a wider scope of changes).

Is NMDA only useful for large/complex routers?

- No, the key premise is that the device returns truthful and accurate information to the best of its ability.
- Hence clients can make decisions based on the **real** device state rather than guessing.
- It should be equally applicable to all situations where YANG is being to used in an automated way.
- For simple devices, it may be trivial to implement (e.g. <operational> matches <running> + config false)

What happens if I cannot align the config and state schema node?

- Up to the modeller.
- Our recommendation is to use two separate leaves, one config true, one config false.
- Try and keep the main operational value on the same path as the configured value if possible.
- E.g. if a setting could be configured statically or negotiated then probably aim to use separate leaves for advertised values, and the same leaf/path for the configured and actual value.

When might intended and operational values deviate?

- Missing/incompatible hardware
- A failure to apply the configuration
- If the operational value has been acquired through some other mechanism:
  - IP addresses/etc from DHCP
  - Protocol timer values from a peer device
  - System created interfaces (Loopback, or hardware based interfaces)
  - System chosen defaults (that are not in the schema)
  - Dynamic configuration set via I2RS
- Latency in applying configuration

What if the “actual” value doesn’t conform to the schema constraints?

- <operational> does not need to be consistent:
- “when”, ”must”, “min-element”, “max-element” statements are not enforced in <operational>, the device should return the truth.
- Syntactic constraints (i.e. hierarchy, identifies, and type constraints are enforced). This is to ensure that a value can be encoded.
- [In future, a generic separate mechanism could be used to report errors on paths in <operational> where no valid value can be returned. Not sure, if this is really required …]

How does the data from dynamic datastores merge with into <operational>

- This must be defined as part of the definition of a dynamic datastore.