Operational State Discussions

2 choices for opstate

1) Adopt the conventions for representing state/config
based on Section 6 of draft-openconfig-netmod-opstate-01

- From a model definition perspective, these conventions impact every model and every model writer.

2) Model OpState using a revised logical datastore definition
as introduced in draft-schoenw-netmod-revised-datastores-00
and also covered in draft-wilton-netmod-refined-datastores-00
There also a variant of this that we believe doesn't significantly
impact this choice.

- With this approach, model definitions need no explicit changes to support applied configuration.

Other drafts impacting the state

- [2] https://tools.ietf.org/html/draft-kwatsen-netmod-opstate-02
- [3] https://tools.ietf.org/html/draft-wilton-netmod-opstate-yang-02

Impact on Ephemeral State (RFC6020bis)

- Constraint enforcement
  - during parsing of RPC payloads (section 8.3.1)
  - during processing of the <edit-config> operation (section 8.3.2)
  - during validation (section 8.3.3)
- If I2RS is only operational state
  - Then selects its own constraint enforcement
  - Models indicate further constraints
- If I2RS configuration is configuration
  - Must adhere to sections 8.3.1, 8.3.2, 8.3.3
  - Deviation per model can be indicated

RFC6020 Constraints (2)

- If a leaf data value does not match the type constraints for the leaf, including those defined in the type's "range", "length", and "pattern" properties, the server MUST reply with an "invalid-value" error-tag in the rpc-error, and with the error- app-tag and error-message associated with the constraint, if any exist.

- If all keys of a list entry are not present, the server MUST reply with a "missing-element" error-tag in the rpc-error.

- If data for more than one case branch of a choice is present, the server MUST reply with a "bad-element" in the rpc-error.

- If data for a node tagged with "if-feature" is present, and the if-feature expression evaluates to "false" in the server, the server MUST reply with an "unknown-element" error-tag in the rpc- error.

Constraints in RFC6020 8.3.1 (2)

- If data for a node tagged with "if-feature" is present, and the if-feature expression evaluates to "false" in the server, the server MUST reply with an "unknown-element" error-tag in the rpc- error.
- If data for a node tagged with "when" is present, and the "when" condition evaluates to "false", the server MUST reply with an "unknown-element" error-tag in the rpc-error.
- For insert handling, if the value for the attributes "before" and "after" are not valid for the type of the appropriate key leafs, the server MUST reply with a "bad-attribute" error-tag in the rpc- error.
- If the attributes "before" and "after" appears in any element that is not a list whose "ordered-by" property is "user", the server

RFC6020bis 8.3.2

- After the incoming data is parsed, the NETCONF server performs the <edit-config> operation by applying the data to the configuration datastore.
  - Delete requests for non-existent data.
  - Create requests for existent data.
  - Insert requests with "before" or "after" parameters that do not exist.
  - Modification requests for nodes tagged with "when", and the "when" condition evaluates to "false". In this case the server MUST reply with an "unknown-element" error-tag in the rpc-error.

RFC6020bis 8.3.3

- When datastore processing is complete, the final contents MUST obey all validation constraints.
  - if the datastore is "running" or "startup", these constraints MUST be enforced at the end of the <edit-config> or <copy-config> operation.
  - If the datastore is "candidate", the constraint enforcement is delayed until a <commit> or <validate> operation.
- Note: I2RS does not fit into this point

Ephemeral State Requirement

Ephemeral-REQ-01

- Ephemeral-REQ-01: I2RS requires ephemeral state; i.e. state that does not persist across reboots. Ephemeral state may consists of ephemeral configured state and operational state. If state must be restored, it should be done solely by replay actions from the I2RS client via the I2RS agent.
- While at first glance this may seem equivalent to the writable- running data store in NETCONF, running-config can be copied to a persistent data store, like startup config. I2RS ephemeral state MUST NOT be persisted.

Constraints on Ephemeral State

- Ephemeral-REQ-02: Non-ephemeral state MUST NOT refer to ephemeral state for constraint purposes; it SHALL be considered a validation error if it does.
- Ephemeral-REQ-03: Ephemeral state must be able to utilized temporary operational state (e.g. MPLS LSP-ID or a BGP IN-RIB) as a constraints.
- Ephemeral-REQ-04: Ephemeral state MAY refer to non-ephemeral state for purposes of implementing constraints

Constraints on Ephemeral State

- Ephemeral-REQ-05: 2RS pub-sub, logging, RPC or other mechanisms may lead to undesirable or unsustainable resource consumption on a system implementing an I2RS Agent. It is RECOMMENDED that mechanisms be made available to permit prioritization of I2RS operations, when appropriate, to permit implementations to shed work load when operating under constrained resources. An example of such a work shedding mechanism is rate-limiting.

  – Note: Ephemeral-REQ-05 is part of the yang-push and event notification technology drafts being worked out in NETCONF WG. It is included here to indicate that these features are necessary for ephemeral state.

Ephemeral State Hierarchy

- Ephemeral-REQ-06: The ability to augment an object with appropriate YANG structures that have the property of being ephemeral. An object defined as any one of the following: yang module, submodule or components of submodule, or schema node.

Local Config vs. Ephemeral state

- Ephemeral-REQ-07: Ephemeral configuration state could override overlapping local configuration state, or vice-versa.
  - Implementations MUST provide a mechanism to choose which takes precedence. This mechanism MUST include local configuration (policy) and MAY be provided via the I2RS protocol mechanisms.

Ephemeral Additions to Yang

- Ephemeral-REQ-08: Yang MUST have a way to indicate in a data model that nodes have the following properties: ephemeral, writable/not- writable, and status/configuration.

Changes to NETCONF

- Ephemeral-REQ-09: The conceptual changes to NETCONF
  - 1. Support for communication mechanisms to enable an I2RS client to determine that an I2RS agent supports the mechanisms needed for I2RS operation.
  - 2. The ephemeral state must support notification of write conflicts using the priority requirements defined in section 7 below in requirements Ephemeral-REQ-11 through Ephemeral-REQ-14).

Changes to RESTCONF

- Ephemeral-REQ-10: The conceptual changes to RESTCONF are:
  - 1. Support for communication mechanisms to enable an I2RS client to determine that an I2RS agent supports the mechanisms needed for I2RS operation.
  - 2. The ephemeral state must support notification of write conflicts using the priority requirements defined in section 7 below in requirements Ephemeral-REQ-11 through Ephemeral-REQ-14).

Multi-headed control (1)

- Ephemeral-REQ-11: The data nodes MAY store I2RS client identity and not the effective priority at the time the data node is stored.
  - Per SEC-REQ-07 in section 3.1 of [I-D.ietf-i2rs-protocol-security-requirements], an identifier must have just one priority. Therefore, the data nodes MAY store I2RS client identity and not the effective priority of the I2RS client at the time the data node is stored. The priority MAY be dynamically changed by AAA, but the exact actions are part of the protocol definition as long as collisions are handled as described in Ephemeral-REQ-12, Ephemeral-REQ-13, and Ephemeral-REQ-14.

Multi-Headed (20

- Ephemeral-REQ-12: When a collision occurs as two clients are trying to write the same data node, this collision is considered an error and priorities were created to give a deterministic result. When there is a collision, a notification MUST BE sent to the original client to give the original client a chance to deal with the issues surrounding the collision. The original client may need to fix their state.

Multi-headed (3)

- Ephemeral-REQ-13: The requirement to support multi-headed control is required for collisions and the priority resolution of collisions. Multi-headed control is not tied to ephemeral state. I2RS is not mandating how AAA supports priority. Mechanisms which prevent collisions of two clients trying the same node of data are the focus.

Multi-headed (4)

- Ephemeral-REQ-14: If two clients have the same priority, the architecture says the first one wins. The I2RS protocol has this requirement to prevent was the oscillation between clients. If one uses the last wins scenario, you may oscillate. That was our opinion, but a design which prevents oscillation is the key point.

Multiple Transations

- Ephemeral-REQ-15: Section 7.9 of the [I-D.ietf-i2rs-architecture] states the I2RS architecture does not include multi-message atomicity and roll-back mechanisms. I2RS notes multiple operations in one or more messages handling can handle errors within the set of operations in many ways. No multi-message commands SHOULD cause errors to be inserted into the I2RS ephemeral data-store.

Ephemeral support in Pub/sub

- Pub-Sub-REQ-01: The Subscription Service MUST support subscriptions against ephemeral data in operational data stores, configuration data stores or both.
- Pub-Sub-REQ-02: The Subscription Service MUST support filtering so that subscribed updates under a target node might publish only ephemeral data in operational data or configuration data, or publish both ephemeral and operational data.
- Note: these are already considered in pub/sub requirements and push data