I2RS Protocol: Requirements + Ideas

Sue Hares
Co-chair summary of requirements
Co-author: Protocol summary
I2RS Requirements for Protocol WG LC

WGLC in October

- draft-ietf-i2rs-ephemeral-state-00
- draft-ietf-i2rs-pub-sub-requirements/
- draft-ietf-i2rs-traceability/
- draft-ietf-i2rs-protocol-security-requirements-01

- Ephemeral State is missing minimum requirements for RESTONF/NETCONF
- My presentation provides background to help NETCONF give I2RS feedback
I2RS Requirements for Protocol WG LC

WGLC after IETF

• draft-ietf-i2rs-security-environment-reqs-01
• Ephemeral State with NETCONF/RESTCONF minimum requirements

Going to IESG with

• Architecture
• Problem statement
• I2RS RIB Information Model
Ephemeral State – 9 requirements

1. Ephemeral state is not unique to I2RS
2. The ephemeral data store is a data store holds configuration that is intended to not survive a reboot.
3. Ephemeral state can be in any data model – so importance of ephemeral is for conformance checking
4. Ephemeral data store is never locked
5. Ephemeral data store can occur in two ways:
   – Yang module that contains both non-ephemeral and ephemeral
   – Yang module that only contains non-ephemeral
   – The yang modules may be protocol modules (BGP) or protocol independent modules (RIB, FB-RIB, Topology)
6. Ephemeral nodes may not have configuration nodes beneath
7. Ephemeral state will be denoted by “ephemeral” in Yang protocol at node level, submodule, or module level
Ephemeral State (4)

8. Caching – is out of scope for the first I2RS protocol release.
   – Long-term concern: latency of I2RS protocol

9. Ephemeral has two error handling extensions
   1. Ephemeral data store allows for reduced error handling that
      MAY remove the requirements for leafref checking, MUST
      clauses, and instance identifier (to allow more speed)
   2. Ephemeral data store allows for priority resolution of write
      operation
      • Priority error resolution means each I2RS client of the ephemeral I2RS
        agent (netconf server) MUST BE associated with a priority.
      • Priority write resolution occurs when a I2RS client with a higher priority
        writes a node which has been written by an I2RS client (with the lower
        priority).
      • When the I2RS agent (netconf server) allows a higher priority client to
        overwrite a lower priority client, the I2RS Agent MAY provide a
        notification indication to entities monitoring the node.

• Agent MUST be able to send notification. Notification can be configured off.

Or
• Agent MUST SEND Notification

Should MAY be MUST?
2 Panes of Glass Model aka (priority resolution)

- Entirely new Nodes
- Groups of Ephemeral Changes
- Single Ephemeral Change
- Constraints

2 Panes of Glass – all or nothing
Types of error checking

• Syntax - correct syntax for node
• Referential – leafref, MUST, instance identifier
• Grouping – group of nodes that should align
  – Stop on error / Continue on error
  assume grouped nodes

• “All or nothing” will be mandated for first pass of I2RS protocol
Other Requirements

• Mutual Authentication based on client identity
  – Client identity passed outside of I2RS (AAA or other)

• Secure transport for Config + other data unless careful designed and reviewed in data model (see connection 2)

• Signaling model capabilities done with Yang library module
Protocol

- **candidate** → **running** → **startup**

- **config true:**
  - **intended config**
  - **applied config**

- **config false:**

**operational data**

Conceptual intended and actual values are determined by the server as an implementation detail.
Thermostat Model

- desired-temp
- intended config
- Actual temp (operational state)

config true;

config false;

Scheduler Client
Simple Thermostat + ephemeral

module thermostat {
  ...
  leaf desired-temp {
    type int32;
    ephemeral true;
    units “degrees Celsius”;
    description “The desired temperature”;
  }

  Operational State:
  leaf actual-temp {
    type int32;
    config false;
    units “degrees Celsius”;
    description “The measured temperature”;
  }
}


Thermostat Model + Hold Temp

- desired-temp
- running datastore
- ephemeral datastore
- config true;
- config false;

- Scheduler Client
- Hold Temp
- intended config
- applied config

Schedule is not deleted by hold-temp button
Desired temperature

Actual temp (operational state)
RESTCONF Example

**RESTCONF Running Datastore Edit**

PUT /restconf/data/thermostat:desired-temp

{ “desired-temp”: 18 }

**RESTCONF Ephemeral Datastore Edit of config=true**

PUT /restconf/data/thermostat:desired-temp?datastore=ephemeral

{ “desired-temp”: 18 }