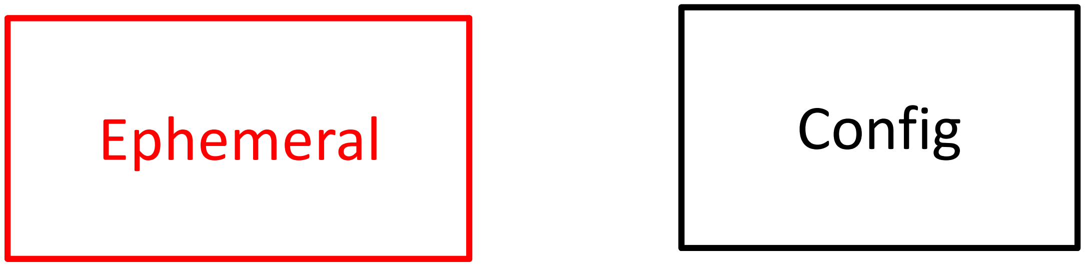
OVERVIEW OF EPHEMERAL STATE ISSUES - AN INTRODUCTION TO EPHEMERAL REQUIREMENTS

Use cases for ephemeral state: Disjoint

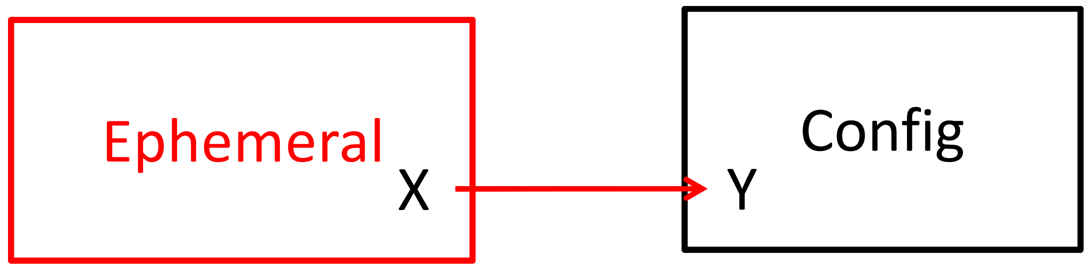
- • Ephemeral state and configuration state do not interact with each other; common protocol operations may retrieve either.



Example use case: A Topology data model that does not use state from other IGP data models

Use cases for ephemeral state: Ephemeral refers to config

- Yang constraints such as “must/when” refer from ephemeral state to config state.
- The reverse is probably never safe. .



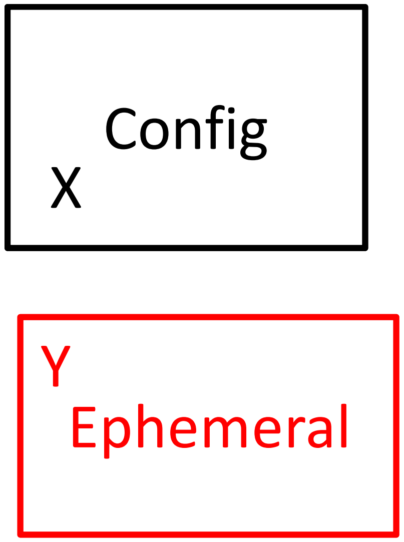
Leaf x has a must relationship on y in the config.
State is otherwise disjoint.

Example use case: A dynamically created BGP neighbor in the Ephemeral datastore uses the Config datastore’s Autonomous System value.

Use cases for ephemeral state: Augmenting

- Rather than "copy and paste" some bit of related config into an i2rs schema, i2rs provides an augmentation on configuration state to provide the i2rs related feature:

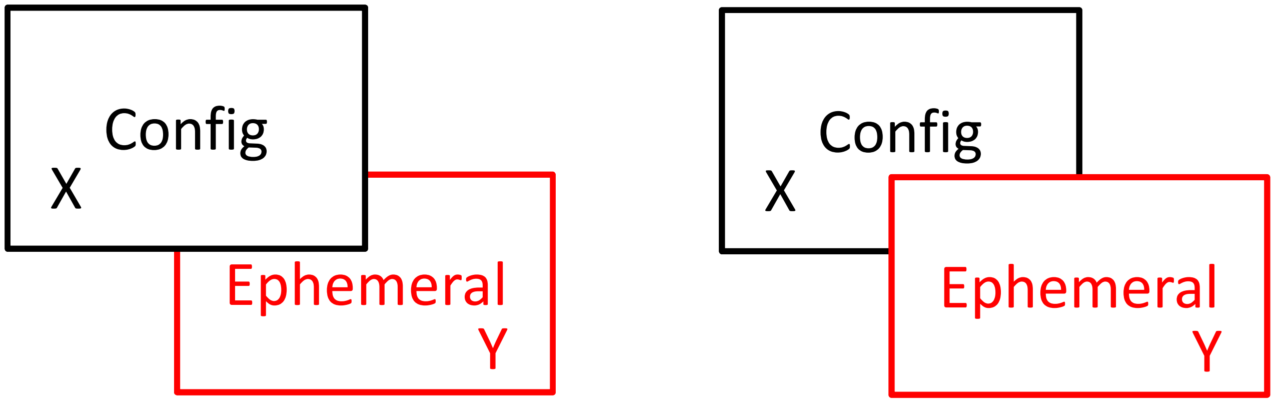
Example use case: An IGP interface has I2RS state adding a "color" Traffic Engineering Attribute



Ephemeral node y is a child of x.

- However, what happens if x is deleted?
- What about consistency between the two separate datastores?

Use cases for ephemeral state: Overriding/occluding



Y could overlap X in different datastores in the same place in the schema. Depending on "priority", a read operation on X may return either the config datastore copy or the ephemeral datastore copy.

Example use case: A static route in the Config's datastore could have its nexthop overridden by dynamic state.

draft-ietf-i2rs-ephemeral-state-reqs-02

Jeffrey Haas
jhaas@pfrc.org

Susan Hares
shares@ndzh.com

Intent of Ephemeral Requirements

- To attempt to provide concrete examples of desired I2RS protocol behavior.
- To drive discussion about potential implementations of that behavior and their representations in
  - I2RS protocol design team
  - netconf/restconf and yang.

Ephemeral state Summary

- 13 Requirements for
  - persistence (1) ,
  - use of constraints with ephemeral (3),
  - changes to yang (1),
  - minimal set of changes to netconf (1)
  - Identifiers (1), Priority & secondary identity (3)
- 7 for PubSub

Persistence

- Ephemeral-REQ-01: I2RS requires ephemeral state; i.e. state that does not persist across reboots. 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 datastore in NETCONF, running-config can be copied to a persistent data store, like startup config. I2RS ephemeral state MUST NOT be persisted.

Constraints used in Ephemeral

- **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 which (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. The designer of ephemeral state modules are advised that such constraints may impact the speed of processing ephemeral state commits and should avoid them when speed is essential.

Changes to Yang

- **Ephemeral-REQ-05:** The ability to add on an object (or a hierarchy of objects) that have the property of being ephemeral. An object needs to be able to have (both)
  - the property of being writable,
  - and the property of the data being ephemeral (or non-ephemeral).

- **Ephemeral-REQ-06**: Yang MUST have a way to indicate in a data model that nodes have the following properties: ephemeral, writable/not-writable, operation state /configuration.

Minimal sub-set of changes

- **Ephemeral-REQ-07:** The minimal set of changes are: (TBD).
  - The minimal set of changes are being discussed in the I2RS protocol design team.

Identifier Requirements

- **Ephemeral-REQ-08:**Clients shall have identifiers, and secondary identifiers.
- Explanation:
  – I2RS requires clients to have an identifier. This identifier will be used by the Agent authentication mechanism over the appropriate protocol.
  – The Secondary identities can be carried as part of RPC or meta-data.
  – The primary purpose of the secondary identity is for traceability information which logs (who modifies certain nodes). This secondary identity is an opaque value.

Priority Requirements

- **Ephemeral-REQ-09:** The data nodes MAY store I2RS client identity and not the effective priority at the time the data node is stored.
  - The I2RS Client MUST have one priority at a time.
  - 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-10, Ephemeral-REQ-11, and Ephemeral-REQ-12.

- **Ephemeral-REQ-10:** 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.

Priority Requirements

- **Ephemeral-REQ-11:** 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.

- **Ephemeral-REQ-12:** 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 WG opinion, but a design which prevents oscillation is the key point.

Transactions

- **Ephemeral-REQ-13**: Section 7.9 of the I2RS architecture document states the I2RS architecture does not include multi-message atomicity and roll-back mechanisms.
  - The I2RS client/agent can send multiple operations within one or more messages.
  - Errors with the set of operations in many message, but No multi-message commands SHOULD cause errors to be inserted into the I2RS ephemeral data-store.

- Error handling techniques
  - Perform all or none
  - Perform until error
  - Perform and store errors and send (as batch) to

Interim note: These revised words are still unclear

Pub/sub Requirements Link to ephemeral

- PubSub-REQ-1: The I2RS interface SHOULD support user subscriptions to data with the following parameters: push of data synchronously or   asynchronously via registered subscriptions.

- PubSSub-REQ-2: Real time for notifications SHOULD be defined by the data models.
- PubSub-REQ-3: Security of the pub/sub data stream SHOULD be able to be model dependent.

- PubSub-REQ-4: The Pub/Sub mechanism SHOULD allow subscription to critical Node Events. Examples of critical node events are BGP peers down or ISIS protocol overload bits.

Pub/sub Requirements Link to ephemeral

- PubSub-REQ-5:I2RS telemetry data for certain protocols (E.g. BGP) will require a hierarchy of filters or XPATHs. The I2RS protocol design MUST balance security against the throughput of the telemetry data.
- PubSub-REQ-6: I2RS Filters SHOULD be able to be dynamic.
- Pub-Sub-REQ-7: I2rs protocol MUST be able to allow I2RS agent to set limits on the data models it will support for pub/sub and within data models to support knobs for maximum frequency or resolution of pub/sub data.

Hierarchy

- Ephemeral configuration may be a child of persistent configuration. The reverse is not permitted.
- Operational state whose parent is ephemeral MUST also be ephemeral.

BACKUP SLIDES

Concrete Proposals made in Past

Flagging configuration state as ephemeral

- Proposal: Extend “config” yang keyword to include “ephemeral”.
- Initial discussion: Consider instead a separate keyword “ephemeral true”.
  - (Martin Bjorkland also points out we’re potentially hitting much of what is in draft-bjorklund-netmod-operational-00.)

Netconf Changes

- Announce an ephemeral-config capability.
- Add a new parameter, “filter-ephemeral” to <get-config> and <get>.
  - Consider alternative from draft-bjorklund-netmod-operational-00. Martin suggests we shouldn’t overload <get-config>.

Secondary identity

- A property of I2RS ephemeral state that is stored for each ephemeral configuration state node.
  - Made accessible to the user as a read-only piece of meta-data. Note that “read-only” meta-data would be a new construct.
  - Carried as a parameter to <edit-config>

Priority

- Similar to secondary-identity, a property of each ephemeral configuration state node.
- User’s priority is assigned as a new attribute of NACM.
- For new ephemeral nodes, it is assigned the user’s priority for that node. (NACM may vary it by path.)
- For existing ephemeral nodes, the update is only permitted if the user’s priority is > the existing node’s priority (First holds) The node then has this priority.
- Transaction/Commit will fail if the user has insufficient priority.
- Presented to the user as read-only meta-data.