I2RS working group
Internet-Draft
Intended status: Standards Track
Expires: September 16, 2016

S. Hares
Huawei

March 15, 2016

I2RS protocol strawman
draft-hares-i2rs-protocol-strawman-00.txt

Abstract

This document provides a strawman proposal for the I2RS protocol covering the ephemeral data store and data flow requirements not covered by i2rs publication/subscription service or traceability. It also proposes additions to YANG for the ephemeral data store and for additional data flow requirements. It proposes additions to the NETCONF and RESTCONF for these additions. Future versions of this document will propose changes to support the I2RS protocol security requirements.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 16, 2016.

Copyright Notice

Table of Contents

1. Introduction

This documents is a strawman for I2RS higher level protocol. The I2RS protocol is a higher level protocol is comprised of a set existing protocols which have been extended to work together to support a new interface to the routing system.

This draft is input to a NETCONF review and design team.

This strawman proposal for the I2RS protocol covers the ephemeral data store and data flow requirements not covered by I2RS publication/subscription service or traceability. It also proposes additions to YANG for the ephemeral data store and for these additional data flow requirements. It also proposes extensions to NETCONF and RESTCONF to support ephemeral state and I2RS.

draft-hares-i2rs-protocol-strawman-examples (pending) provides examples of this strawman protocol use for I2RS. This draft uses a simple thermostat model to illustrate commands.

1.1. Ephemeral Changes

This document proposes additions to support ephemeral state in the datastores supported by NETCONF and RESTCONF, and in the YANG modules that define these data stores. The requirements for the I2RS ephemeral state are covered in [I-D.ietf-i2rs-ephemeral-state]

This draft provides suggests the following additions to support the I2RS ephemeral state:

- Yang ephemeral statement,
- NETCONF ([RFC6241]) protocol extensions for the ephemeral data store,
- RESTCONF ([I-D.ietf-netconf-restconf]) protocol extensions for the ephemeral data store

1.2. Data Flow Changes

This document proposes additions to support data flows from different data models for large data flows, traffic monitoring, actions and OAM interaction, and flows during outages or attacks. The requirements for these changes are define in [I-D.hares-i2rs-dataflow-req].

Most large data flows will be handled utilizing the publication/ subscription service define in the I2RS publication/subscription service requirements specified in [I-D.ietf-i2rs-pub-sub-requirements]. Extensions to NETCONF to support a push publication/subscription service have been defined in [I-D.ietf-netconf-yang-push]. This document does not propose a pull publication/subscription (pull pub-sub) service for the first set of component protocols for the I2RS higher level protocol. If deployments require the pull pub-sub service, then an expansion of the push service can provide one mechanism.

This document does provide support for the I2RS protocol to:

Support large data transfers in a data agnostic format (DF-REQ-02) supporting transfers of data in any format (E.g. XML, JSON, MTL, protobuf, ASCII) over any transport (DF-REQ-03).

Support the use of IPFIX as a component protocol to send traffic monitoring data or any type of large data flow from I2RS agent to I2RS client (DF-REQ-04),

Support exporting traffic statistics for filter-based policies usage (BGP-FS, I2RS FB-FIB, policy routing), IPPM, SFLOW and other

traffic statistics using either yang models or IPFIX template formats over any data encapsulation format over any transport (DF-REQ-05).

2. Definitions Related to Ephemeral Configuration

Currently the configuration systems managed by NETCONF ([RFC6241]) or RESTCONF ([I-D.ietf-netconf-restconf]) have three types of configuration: candidate, running, and startup running under the config=true flag.

- The candidate receives configuration changes from NETCONF/ RESTCONF.
- The running configuration is the configuration currently operating on a devices
- The start-up configuration is the configuration that survives a reboot.

The config=false flag has operational data which exists alongside the config=true data. However, at this point there is no datastored defined for configuration false.

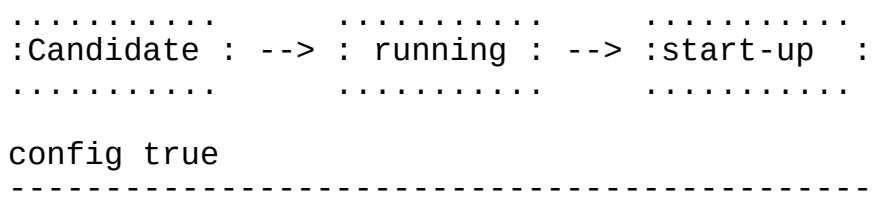

Figure 1

The [I-D.ietf-netmod-opstate-reqs] defines new terms to clarify how this works. In reality, the running configuration becomes the intended configuration that is intended to be loaded into a device. The loading of the update into the system can be either asynchronous or synchronous. In the asynchronous case, the netconf server responds to the client after the intended has been updated, but the applied configuration is only updated later when the configuration change has full impacted all components on the device. The synchronous configuration operation occurs when both the intent configuration has been updated and the actual configuration has been loaded after resolving the necessary things to load in a box.

This document will use the terms defined in [I-D.ietf-netmod-opstate-reqs].

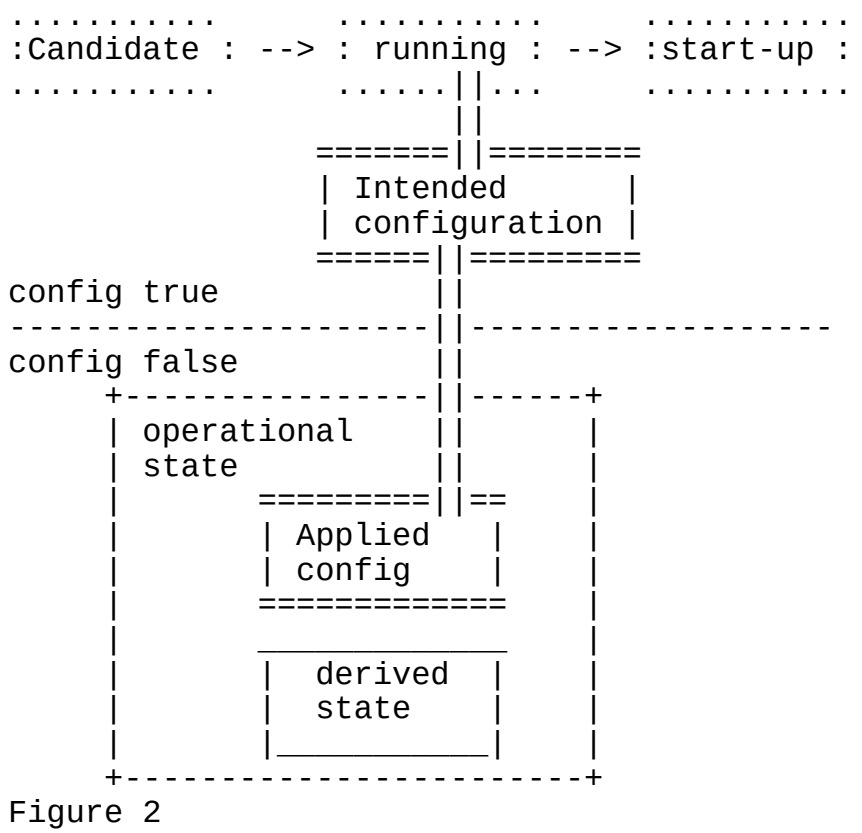

Figure 2

3. Definition of ephemeral datastore for NETCONF/RESTCONF

This section describes the properties of the ephemeral datastore. The ephemeral datastore is not unique to I2RS. This approach to the ephemeral datastore is a panes-of-glass model. This definition of I2RS does not support caching in the I2RS Agents. Future I2RS work may reconsidered supporting caching.

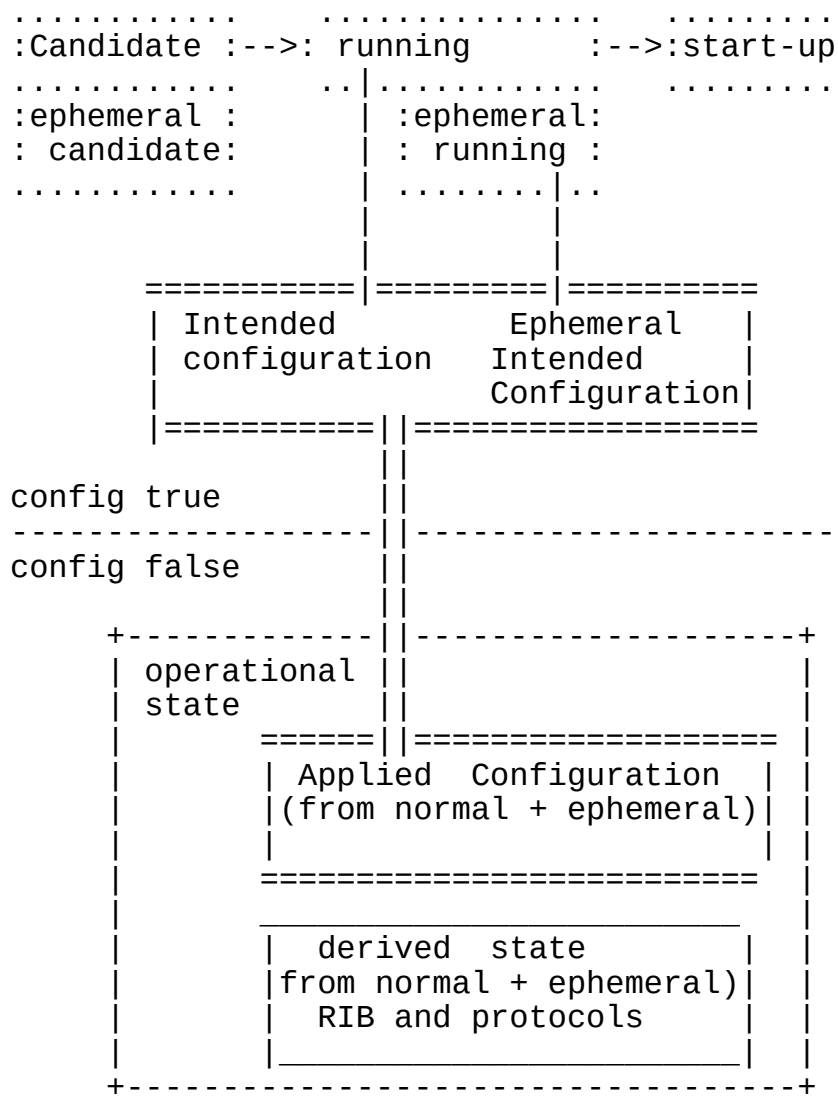

Figure 3

The ephemeral data store has the following qualities:

1. Ephemeral state is not unique to I2RS work.

2. The ephemeral datastore is never locked.

3. The ephemeral datastore is really a portion of the candidate, running, and intended configuration that does not persist over a reboot.

   * Since Ephemeral is just about data not presisting over a reboot, then in theory any node or group of nodes in a YANG data model could be ephemeral. The YANG data module must indicate what portion of the data model (if any) is ephemeral.

* A YANG data module could be all ephemeral (e.g. [I-D.ietf-i2rs-rib-data-model]) with no directly associated configuration models,

* A YANG model could be all ephemeral but associated with a configuration model (E.g. (draft-hares-i2rs-bgp-dm-01),

* or a single data node or data tree could be made ephemeral.

4. The applied configuration is the result of the the intent configuration (normal and ephemeral). Similarly, the derived data is a result of the applied configuration.

5. Ephemeral portions (node, tree, or data model) need to be signalled in the conformance portions of the NETCONF and RESTCONF work. Conformance is signalled in the following ways:

   * The conformance portion of NETCONF ([RFC6241]) is currently signalled in the <hello>.

   * Yang 1.1 and RESTCONF uses the module library ([I-D.ietf-netconf-yang-library])

   * NETCONF may use the module library in the future.

   * The ephemeral status in a module will be listed as "all, none or partial". Optionally the module may provide a list of nodes.

6. The ephemeral data store is treated as one pane of glass that an I2RS client(s) may read/write which has the following implications:

   * The ephemeral datastore overlays the configuration datastore at each point it touches (candidate, running, and intended configuration) By overlays, the I2RS write overwrites a previous configuration value, but if a local configuration value changes after that over-write the default is to have the local-config win. [aka Last Write wins.]

     + An example may help to illustrate this default rule. Say a configuration specifies a local route of 128.2/16 with a nexthop of 192.5.10.1. Afterwards an ephemeral route is added for 128.2/16 with nexthop of 192.5.10.2. This ephemeral route would replace the first route. If the configuration changes the underlying route (128.2/16 with nexthop of 192.5.10.1) and the default rule of local configuration is in effect, the local configuration value

(128.2/16 with nexthop of 192.5.10.1) would take effect. This follows the normal netconf concept that Last configured wins. The I2RS agent would notify the I2RS Client that the ephemeral route (128.2/16 with nexthop of 192.5.10.2) had been overwritten by the local configuration.

* The default of local can be changed by operator-applied policy to allow ephemeral to always win or local configuration to always win, but the status of the operator applied policy must be queryable in the I2RS agent (if that scope) or in the I2RS ephemeral data model. I2RS clients are required to understand and handle if the an I2RS agent supports something different than the default (aka Last write wins).

4. Error handling

Ephemeral nodes level of validation/error handling in the I2RS protocol has three following three types:

- syntax validation only,
- Ephemeral data store allows for reduced error handling that removes the requirements for referential checks [I2RS normal error handling]
- ephemeral data store handling that uses normal NETCONF/RESTCONF error handling with syntax and referential [full],

The default error handling is "no referential checking".

Normal error handling of I2RS Agent for an I2RS client’s information examines the following:

- message syntax validation,
- syntax validation for nodes of data model,
- removes referential requirements for leafref checking, MUST clauses, and instance indentifier,
- grouping of data within a data models or across data models to accomplish tasks,
- permission to write nodes of data model,
- grouping,

- priority to write nodes of data model being higher than existing priority

The full error handling status includes all checks included for any normal YANG data module uses by NETCONF/RESTCONF. This includes referential checks for leafref checks, MUST clauses, and instance identifiers.

The I2RS Data model sets permissible range of error handling for writes on a data model (none, I2RS normal, full), and this may be further restricted by an operator applied policy. If an I2RS client requests a lower level of error handling that permissible, the I2RS Agent will return an error.

Multiple I2RS clients writing to the same variable is considered an "error condition" in the I2RS architecture [I-D.ietf-i2rs-architecture], but the I2RS Agent must handle this error condition. Upon multiple I2RS clients writing, the ephemeral data store allows for priority pre-emption of the write operation. Priority pre-emption means each I2RS client of the ephemeral I2RS agent (netconf server) is associated with a priority. Priority pre-emption occurs when a I2RS client with a higher priority writes a node which has been written by an I2RS client (with the lower priority). At this point, the I2RS agent (netconf server) allows the write and provides a notification indication to the notification publication/subscription service.

This section describes the ephemeral data stores handling of each of these error functions.

4.1. Syntax validation

Syntax validation of the message should be done according to the NETCONF or RESTCONF protocol features. New features for ephemeral datastore should provide the error handling with the feature. Message syntax validation can be for read, write, or rpc functions.

Syntax validation of the data model included in the ephemeral data store should be done by I2RS Agent.

4.2. Referential validation

The ephemeral data store normal processing does not do the following referencial checks: leafref, MUST, instance identifier. The removal of these validations allows for intelligent I2RS clients to rapidly read or write data, and handle error conditions at a higher level.

4.3. Grouping and Error handling

Yang 1.0 and Yang 1.1 provide the ability to group data in groupings, leafref lists, lists, and containers. Data model group data in order to group data that is logically associated with one another. Data models may logical group data across groupings. One example of such an association is the association of a static route with an interface. The concepts of groupings apply to both ephemeral and non-ephemeral nodes within a data model.

4.3.1. Initial Support of Parital Writes

The initial releases of I2RS will only require "all-or-nothing" in the I2RS Agent.

4.3.2. Future Scope of multiple message writes

Error handling on writes of the ephemeral datastore is different for nodes that are grouped versus orthogonal. Group nodes may need to be all changed or all removed (all-or-nothing). In contrast, writing orthogonal data nodes in the same data module or between data models need to be added or deleted in sync.

The [I-D.ietf-i2rs-architecture] specifies three types of error handling for a partial write operation: "all-or-nothing", "stop-on-error", or "continue-on-error". Partial write operations of "stop-on-error" or "continue-on-error" are allowed only for data writes which are not a part of a grouping within a data model. The definition of the I2RS error conditions are:

- stop-on-error - means that the configuration process stops when a write to the configuration detects an error due to write conflict.

- continue-on-error - means the configuration process continues when a write to the configuration detects an error due to write process, and error reports are transmitted back to the client writing the error.

- all-or-nothing - means that all of the configuration process is correctly applied or no configuration process is applied. (Inherent in all-or-nothing is the concept of checking all changes before applying.)

4.3.2.1. NETCONF Support of Partial Writes

NETCONF does not support a mandated sequencing of edit functions or write functions. Without this mandated sequences, NETCONF cannot support partial edits.

4.3.2.2. RESTCONF Support of Partial Writes

RESTCONF has a complete set of operations per message. The RESTCONF patch can support write functions per messages.

4.4. priority preemption

I2RS protocol uses priority to resolve two I2RS clients having permissions to write the same pieces of data in an I2RS agent (NETCONF server). If two (or more) I2RS clients attempt to write the same data, the the one with the highest priority is enable to write the data. In the case of two clients with teh sample priority attempting to write data, the the first one to request write wins.

Each client has a unique priority. Client identities and priorities are assigned outside of I2RS by exterior mechanisms such as AAA or adminstrative interfaces. A valid I2RS client must have both an identity and a priority.

A client-id and priority must be saved per node.

A sample container for I2RS client information is shown below.

```
   container i2rs-clients {
       leaf max-clients {
          config false;
          mandatory true;
          type uint32 {
            range "1 .. max";
          }
       }
       list i2rs-client {
          key name;
          unique priority;
          leaf name { ... }
          leaf priority { ... }
       }
    }
```

Figure 4

4.4.1. Andy Bierman Priority Comment

(Andy)The priority is not required to be densely numbered. Whether there are 1 pane per client or 1 pane per priority or 1 giant blob full of everything, the code will be the same. The goal of "unique priority" is to require that only priority be saved in the meta-data for the ephemeral datastore. Without that, client-id and priority must be saved (per data node).

5. transport protocol

5.1. Secure Protocols

NETCONF's XML-based protocol ([RFC6241]) can operate over the following secure and encrypted transport layer protocols:

SSH as defined in [RFC6242],

TLS with X.509 authentication [RFC7589]

RESTCONF's XML-based or JSON [RFC7158] data encodings of Yang functions are passed over http with (GET, POST, PUT, PATCH, DELETE, OPTIONS, and HEAD).

5.2. Insecure Protocol

The ephemeral database may support insecure protocols for information which is ephemeral state which does not engage in configuration. The insecure protocol must be defined in conjunction with a data model or a subdata model.

6. Yang Library Use by Ephemeral

The data modules supporting the ephemeral datastore can use the Yang module library to describe their datastore. Figure 5 shows the module library data structure as found [I-D.ietf-netconf-yang-library]. The I2RS modules will provide features for I2RS ephemeral state and protocol of:

- protocol version support - "version 1",
- ephemeral model scope - ephemeral modules, mixed config module (ephemeral and config), mixed derived state (ephemeral and config).
- ephemeral checking - syntax only, I2RS normal (no referential), full (RESTCONF/NETCONF)
- multiple message support - "all or nothing",
- pane of glass support - "single only".
- protocol supported - "NETCONF", "RESTCONF", "NETCONF pub-sub push",
- encoding support - XML or JSON

o transports supported: "TCP", "SSH", "TLS", and others supported.

```
  +--ro modules
     +--ro module*[name revision]
        +--ro name  yang: yang-identifier
        +--ro revision  union;
        +--ro schema?   inet:uri
        +--ro namespace   inet:uri
        +--ro feature*    yang:yang-identifier
        +--ro deviation* [name revision]
        |  +-- ro name   yang:yang-identifier
        |  +-- ro revision union
        +--ro conformance enumeration
        +--ro submodules
           +--ro submodule*[name revision]
              +--ro name yang:yang-identifier
              +--ro revision  union
              +--ro schema?  inet:uri
```

Figure 5

7. Simple Thermostat Model

In this discussion of ephemeral configuration, this draft utilizes a simple thermostat model with the YANG configuration found in figure 6.

```
module thermostat {
  ..
  leaf desired-temp {
     type int32;
         units "degrees Celsius";
         description "The desired temperature";
         }

  leaf actual-temp {
     type int32;
         config false;
         units "degrees Celsius";
         description "The measured temperature
         (operational state).";
         }
  }
```

Figure 6 - Simple thermostat model yabng

Figure 6 shows two I2RS clients talking to this model: scheduler and hold-temp. Scheduler has a schedule set of temperatures to put in

the thermostat. Hold-temp holds the temperature at the same value. The hold-temp I2RS client has a higher priority than the scheduler client.

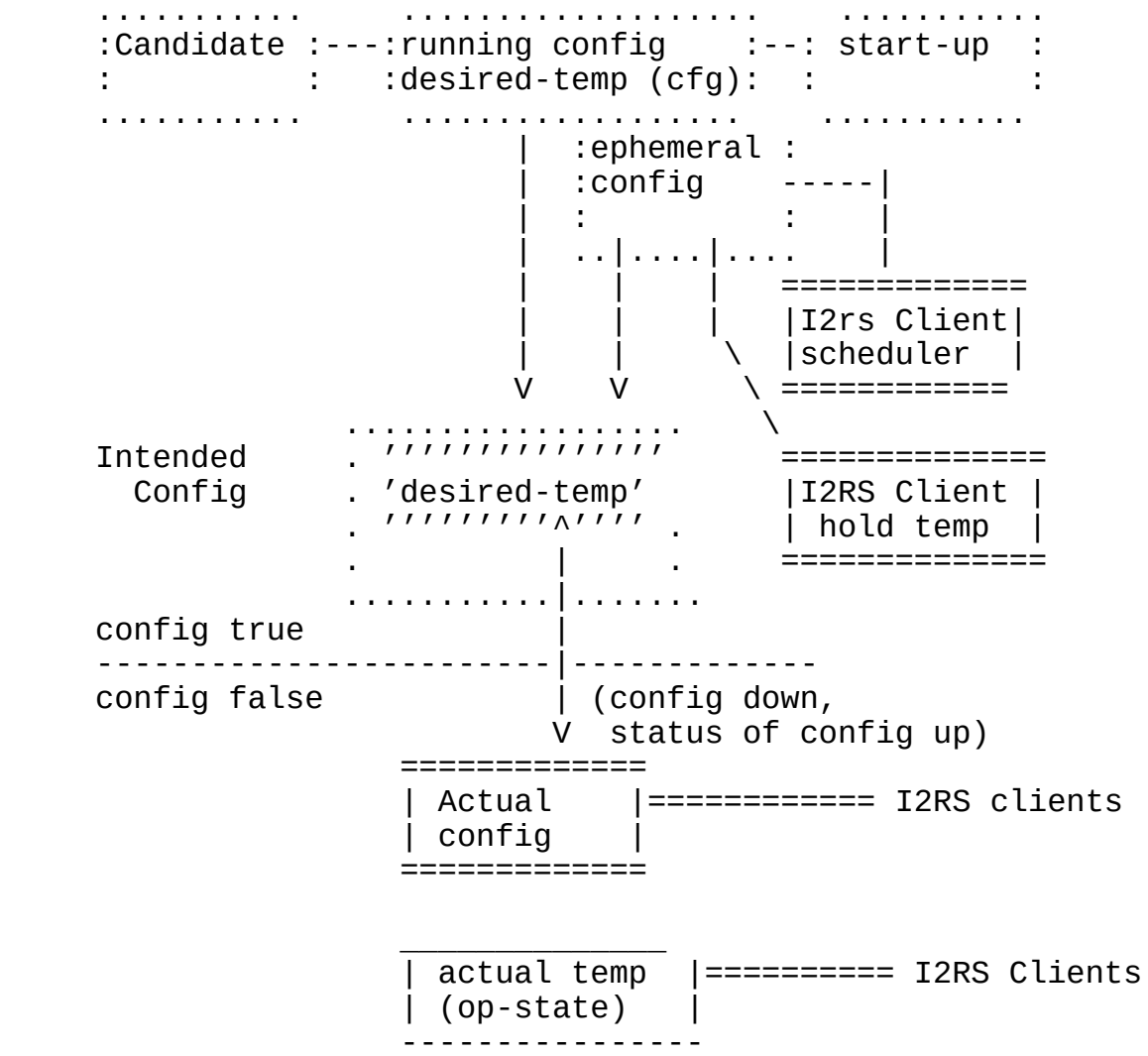

Figure 6 - Two I2RS clients

7.1. Yang changes

Yang needs to adds key word ephemeral with the following key words:

- full-check - defined as NETCONF/RESTCONF relevant error checks,
- I2RS checks - defined as no referential checks (see section above),
- syntax only - defined as only message syntax and model syntax,

```
module thermostat {
  ..

  leaf desired-temp {
     type int32;
         units "degrees Celsius";
         ephemeral true;
         ephemeral-validation full-check;
         description "The desired temperature";
         }

  leaf actual-temp {
     type int32;
         config false;
         units "degrees Celsius";
         description "The measured temperature";
         }
  }
```

Figure 7 - Simple Thermostat Yang with ephemeral

7.2. RESTCONF sequence

Figure 7 shows the thermostat model has ephemeral variable desired-temp in the running configuration and the ephemeral data store. The RESTCONF way of addressing is below:

RESTCONF running data store

```
PUT /restconf/data/thermostat:desired-temp
{"desired-temp":18}
```

RESTCONF ephemeral datastore

```
PUT /restconf/data/thermostat:desired-temp?datastore=ephemeral
?ephemeral-validation="full-check"
{"desired-temp":19 }
```

Figure 8 - RESTCONF setting of ephemeral state

7.3. NETCONF messages

The NETCONF way of transmitting this data would be

```
<rpc-message-id=101
  xmlns="urn:ietf:params:xml:ns:base:1.0">
  <edit-config>
    <target>
     <ephemeral >
            true
        </ephemeral >
        <ephemeral-validation>
           fullcheck
        </ephemeral-validatio>
    </target>
    <config>
      <top xmlsns="http:://example.com/schema/1.0/thermostat/config>
       <desired-temp> 18 </desired-temp>
      </top>
    </config>
   </edit-config>
</rpc>
```

Note: config=TRUE; datastore = ephemeral
ephemeral-validation=full-check;

figure 8 NETCONF setting of desired-temp

8. NETCONF protocol extensions for the ephemeral datastore

capability-name: ephemeral-datastore

8.1. Overview

This capability defines the NETCONF protocol extensions for the ephemeral state. The ephemeral state has the following features:

- the ephemeral datastore is a datastore holds configuration information (Config=true) that is intended to not survive a reboot.
- The ephemeral capbility is signalled as a capability for a node, a sub-module, or a module either in the conformance portion of NETCONF (<hello>) or via netconf yang module library ([I-D.ietf-netconf-yang-library]) used by Yang 1.1 and RESTCONF.
- ephemeral data will be noted by an "ephemeral statement at the node or module "
- The ephemeral datastore is never locked.

- The ephemeral data store is one pane of glass that overrides the intended config which is normally the running datastore, but can be designated as the candidate config.

- Ephemeral data nodes can occur as part of protocol or protocol independent modules. However, ephemeral data nodes cannot have non-ephemeral data nodes within the subtree. Ephemeral sub-modules cannot have non-ephemeral data nodes wihin the module. Ephemeral modules cannot have non-ephemeral sub-modules or nodes within the module.

- ephemeral writes have two checks: error validation and priority premption between two I2RS client writes to the same data.

- ephemeral error checking has the following three levels

  * syntax only - message and data module syntax,

  * reduced error checking that remove the requirement for leafref checking, MUST clauses, and instance identifier validation.

  The default is reduced error checking.

- The write operation with a priority pre-emption by a higher priority client of the lower priority clients write where the overwrite triggers a notification by the I2RS agent to the lower priority client.

8.2. Dependencies

The following are the dependencies for ephemeral support:

The Yang data modules must be flag with the ephemeral data store at the node, sub-module and model.

the Yang data models must specify ephemeral validation if the models desire validation other reduced error checking.

The Yang modules must support the notification of write-conflicts.

8.3. Capability identifier

The ephemeral-datastore capability is identified by the following capability string: (capability uri)

8.4. New Operations

8.4.1. link-ephemeral

The <link-ephemeral> allows the ephemeral datastore to be a pane of glass that impacts either the running-config configuration pane of glass or the candidate configuration pane of glass.

<link-ephemeral> target-config

where target config is:
writable-running or candidate config.

8.4.2. Bulk-Write

Bulk Write allows for large scale writes with error handling that is specified as syntax or reduced or full. Alternatively, the data modules can utilize an RPC to do bulk reads/writes. The bulk write will be first check for other I2RS clients having a higher priority write value for any of the values.

Editor: Do we need something beyond rpc for bulk data writes?

8.5. Modification to existing operations

The capability for :ephemeral-datastore modifies the target for existing operations.

8.5.1. <get-config>

The :ephemeral-datastore capability modifies the <edit-config> to accept the <ephemeral> as a target for source, and allows the filters focused on a particular module, submodule, or node.

The positive and negative responses remain the same.

```
   Example - retrieve users subtree from
             ephemeral database

    <rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <get-config>
         <source>
           <emphemeral-datastore/>
         </source>
         <filter type="subtree">
           <top xmlns="http://example.com/schema/1.0/thermostat/config">
           <desired-temp>
            </top>
         </filter>
      </get-config>
    </rpc>
```

8.5.2. <edit-config>

The :ephemeral-datastore capability modifies the <edit-config> to accept the <ephemeral> as a target for source with filters. The operations of merge, replace, create, delete, and remove are available, but each of these operations is modified by the priority write as follows:

<merge> parameter is replaced by <merge-priority> The current data is modified by the new data in a merge fashion only if existing data either does not exist, or is owned by a lower priority client. If any data is replaced, this event is passed to the notification function within the pub/sub and traceability.

<replace> is replaced by <replace-priority> for ephemeral datastore which replaces data if the existing data is owned by a lower priority client. If data any data is replaced, this event is passed to the notification function within pub/sub and traceability for notification to the previous client. The success or failure of the event is passed to traceabilty.

<create> - the creation of the data node works as in [RFC6241] except that the success or failure is passed to pub/sub and traceability functions.

<deletion> - the deletion of the data node works as in [RFC6241] except event that the success or the error event is passed to the notiication function withi pub/sub and traceability functions.

<remove> - the remove of the data node works as in [RFC6241] except that all results are forwarded to traceabilty.

The existing parameters are modified as follows:

<target> - add a target of :emphemeral-datastore

<default-operation> -allows only <merge-priority> or <replace-priority>

<error-option> - the I2RS agent agent supports only the a"all-or-nothing" equivalent to a "rollback-on-error" function.

positive response - the <ok> is sent for a positive response within an <rpc-reply>.

negative response - the <rpc-error> is sent for a negative response within an <rpc-reply>. Note a negative respones may evoke a publication of an event.

8.5.3. <copy-config>

Copy config allows for the complete replacement of all the ephemeral nodes within a target. The alternation is that source is the :ephemeral datastore with the filtering to match the datastore. The following existing parameters are modified as follows:

<target> - add a target of :emphemeral-datastore

<error-option> - the I2RS agent agent supports only the a"all-or-nothing" equivalent to a "rollback-on-error" function.

positive response - the <ok> is sent for a positive response within an <rpc-reply>.

negative response - the <rpc-error> is sent for a negative response within an <rpc-reply>.

8.5.4. <delete-config>

The delete will delete all ephemeral nodes out of a datastore. The target parameter must be changed to allow :ephemeral-datastore. and filters.

8.5.5. <lock> and <unlock>

Lock and unlock are not supported with a target of :ephemeral-datastore.

8.5.6. <get>

The <get> is altered to allow a target of :ephemeral-datastore and with the filters.

8.5.7. <close-session> and <kill-session>

The close session is modified to take a target of :ephemeral-datastore Since no locks are set, none should be released.

The kill session is modified to take a target of "ephemeral-datastore. Since no locks are set, none should be released.

8.6. Interactions with Capabilities

[RFC6241] defines NETCONF capabilities for writeable-running datastore, candidate config data store, confirmed commit, rollback-on-error, validate, distinct start-up, URL capability, and XPATH capability.

8.6.1. writable-running and candidate datastore

The writeable-running and the candidate datastore can be used in conjunction with the ephemeral data store. Ephemeral database overlays an intended configuration, and may overlay candidate and running configuration data store. The <link-ephemeral> operation links to these two databases.

8.6.2. confirmed commmit

Confirmed commit capability is not supported for the ephemeral datastore.

8.6.3. rollback-on-error

The rollback-on-error when included with ephemeral state allows the error handling to be "all-or-nothing" (roll-back-on-error).

8.6.4. validate

The <validate> key word is expanded to support the following:

source: ephemeral-datastore

validate: (syntax, no-referential, full) with the following definitions:

* syntax - validates only the message syntax and the data base syntax.

* no-referentail - skips referential test (leafref, MUST clauses, and instance identifiers).

* full - all normal netconf/restconf module error chcking

8.6.5. Distinct Startup Capability

This NETCONF capability appears to operate to load write-able running config, running-config, or candidate datastore. The ephemeral state does not change the environment based on this command.

8.6.6. URL capability and XPATH capability

The URL capabilities specify a <url> in the <source> and <target>. The initial suggestion to allow both of these features to work with ephemeral operation.

9. RESTCONF protocol extensions for the ephemeral datastore

capability-name: ephemeral-datastore

9.1. Overview

This capability defines the RESTCONF protocol extensions for the ephemeral state. The ephemeral state has the features described in the previous section on NETCONF.

9.2. Dependencies

The ephemeral capabilities have the following dependencies:

Yang data nodes, sub-modules, or modules must be flaged with the config datastore flag;

The Yang modules must support the notification of write-conflicts.

The I2RS Yang modules must support the following:

the yang-patch features as specified in [I-D.ietf-netconf-yang-patch].

The yang module library feature [I-D.ietf-netconf-yang-library],

9.3. Capability identifier

The ephemeral-datastore capability is identified by the following capability string: (capability uri)

9.4. New Operations

none

9.5. modification to data resources

RESTCONF must be able to support the ephemeral datstore with its rules as part of the "{+restconf}/data" subtree. The "edit collision" features in RESTCONF must be able to provide notification to I2RS read functions or to rpc functions. The "timestamp" with a last modified features must support the traceability function.

The "Entity Tag" could support saving a client-priority tuple as a opaque string, but it is important that that additions be made to restore client-priority so it can be compared with stromgs to be added.

9.6. Modification to existing operations

The current operations in RESTCONF are: OPTIONS, HEAD, GET, POST, PUT, PATCH, and DELETE. This section describes the modification to these exiting operations.

9.6.1. OPTIONS changes

The options methods should be augmented by the [I-D.ietf-netconf-yang-library] information that will provide an indication of what ephemeral state exists in a data modules, or a data modules sub-modules or nodes.

9.6.2. HEAD changes

The HEAD in retrieving the headers of a resources. It would be useful to changes these headers to indicate the datastore a node or submodule or module is in (ephemeral or normal), and allow filtering on ephemeral nodes or trees, submodules or module.

9.6.3. GET changes

GET must be able to read from the URL and a particular datastore. Similarly, it is important the Get be able to determine if it is a ephemeral data store.

9.6.4. POST changes

POST must simply be able to create resources in ephemeral datastores ("=datastore=ephemeral") and invoke operations defined in ephemeral data models using the rules of the ephemeral database.

9.6.5. PUT changes

PUT must be able to reference an ephemeral module, sub-module, and nodes ("?datastore=ephemeral").

9.6.6. PATCH changes

Plain PATCH must be able to update or create child resources in an ephemeral datastore ("?datstore=ephemeral") The PATCH for the ephemeral state must be change to provide a merge or update of the original data only if the client’s using the patch has a higher priority than an existing datastore’s client, or if PATCH requests to create a new node, sub-module or module in the datastore.

9.6.7. DELETE changes

The phrase "?datastore=ephemeral" following an element will specify the ephemeral data store when deleting entry.

9.6.8. Query Parameters

The query parameters (content, depth, fields, insert, point, start-time, stop-time, and with-defaults (report-all, trim, explicit, report-all-tagged) must support ephemeral datastores ("?datastore=ephemeral") described above.

9.7. Interactions with Notifications

The ephemeral database must support subscribing to receiving notifications as Event stream. The event error stream processing should support the publication/subscription mechanisms for ephemeral state defined in [I-D.ietf-netconf-yang-push].

9.8. Interactions with Error Reporting

The ephemeral database] support in RESTCONF must also support passing error information regarding ephemeral data access over to RESTCONF equivalent of the and traceability client.

10. IANA Considerations

This is a protocol strawman - nothing is going to IANA.

11. Security Considerations

The security requirements for the I2RS protocol are covered in [I-D.ietf-i2rs-protocol-security-requirements]. The security environment the I2RS protocol is covered in [I-D.ietf-i2rs-security-environment-reqs]. Any person implementing or deploying the I2RS protocol should consider both security requirements.

12. Acknowledgements

This document is an attempt to distill lengthy conversations on the I2RS proto design team from August

Here’s the list of the I2RS protocol design team members

- Alia Atlas
- Ignas Bagdonas
- Andy Bierman
- Alex Clemm
- Eric Voit
- Kent Watsen
- Jeff Haas
- Keyur Patel
- Hariharan Ananthakrishnan
- Dean Bogdanavich
- Anu Nair
- Juergen Schoenwaelder
- Kent Watsen

13. Major Contributors

- Andy Bierman (Yuman Networks) - andy@yumaworks.com
- Kent Watson (Juniper) (kwatsent@juniper.net)

14. References

14.1. Normative References:

[I-D.hares-i2rs-dataflow-req]
Hares, S., "I2RS Data Flow Requirements", draft-hares-i2rs-dataflow-req-01 (work in progress), March 2016.

[I-D.ietf-i2rs-architecture]
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-13 (work in progress), February 2016.

[I-D.ietf-i2rs-ephemeral-state]
Haas, J. and S. Hares, "I2RS Ephemeral State Requirements", draft-ietf-i2rs-ephemeral-state-04 (work in progress), March 2016.

[I-D.ietf-i2rs-protocol-security-requirements]
Hares, S., Migault, D., and J. Halpern, "I2RS Security Related Requirements", draft-ietf-i2rs-protocol-security-requirements-03 (work in progress), March 2016.

[I-D.ietf-i2rs-pub-sub-requirements]
Voit, E., Clemm, A., and A. Prieto, "Requirements for Subscription to YANG Datastores", draft-ietf-i2rs-pub-sub-requirements-05 (work in progress), February 2016.

[I-D.ietf-i2rs-rib-data-model]
Wang, L., Ananthakrishnan, H., Chen, M., amit.dass@ericsson.com, a., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-04 (work in progress), November 2015.

[I-D.ietf-i2rs-rib-info-model]
Bahadur, N., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-08 (work in progress), October 2015.

[I-D.ietf-i2rs-security-environment-reqs]
Migault, D., Halpern, J., and S. Hares, "I2RS Environment Security Requirements", draft-ietf-i2rs-security-environment-reqs-00 (work in progress), October 2015.

[I-D.ietf-i2rs-traceability]
Clarke, J., Salgueiro, G., and C. Pignataro, "Interface to the Routing System (I2RS) Traceability: Framework and Information Model", draft-ietf-i2rs-traceability-07 (work in progress), February 2016.

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-09 (work in progress), December 2015.

[I-D.ietf-netconf-yang-library]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", draft-ietf-netconf-yang-library-04 (work in progress), February 2016.

[I-D.ietf-netconf-yang-patch]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Patch Media Type", draft-ietf-netconf-yang-patch-07 (work in progress), December 2015.

[I-D.ietf-netconf-yang-push]
Clemm, A., Prieto, A., Voit, E., Tripathy, A., and E. Einar, "Subscribing to YANG datastore push updates", draft-ietf-netconf-yang-push-01 (work in progress), February 2016.

[I-D.ietf-netmod-opstate-reqs]
Watsen, K. and T. Nadeau, "Terminology and Requirements for Enhanced Handling of Operational State", draft-ietf-netmod-opstate-reqs-04 (work in progress), January 2016.

[I-D.ietf-netmod-yang-metadata]
Lhotka, L., "Defining and Using Metadata with YANG", draft-ietf-netmod-yang-metadata-06 (work in progress), March 2016.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://www.rfc-editor.org/info/rfc6242>.

[RFC7158] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7158, DOI 10.17487/RFC7158, March 2014, <http://www.rfc-editor.org/info/rfc7158>.

[RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <http://www.rfc-editor.org/info/rfc7589>.

14.2. Informative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

Author’s Address

Susan Hares
Huawei
Saline
US

Email: shares@ndzh.com