I2RS working group
Internet-Draft
Intended status: Standards Track
Expires: April 14, 2016

S. Hares
Huawei
A. Beirman
YumaWorks
K. Watsen
Juniper

October 12, 2015

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

Abstract

This document provides a strawman proposal for the I2RS protocol covering the ephemeral data store. It provides Yang ephemeral statement, netconf protocol extensions for the ephemeral data store, and RESTCONF protocol extensions for the protocol data store.

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 April 14, 2016.

Table of Contents

1. Introduction

This documents is a strawman for the I2RS Protocol from early I2RS design team discusses. It focuses on the protocol extensions for ephemeral data store.

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

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

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

2. Resolve before publishing draft

1. (dean)What edits are allowed in the ephemeral data store. Should those be syntactically correct or syntacticly and semantically?

2. Validation slows things down a lot, so datastore validation needs to be considered carefully. This is where I think routing expertise will help decide how much validation can really be skipped for a particular use-case.

3. (Anu)On priority: So the priority maximum will be same as the max-clients , if we are assigning unique priorities from 1 - max-clilents. (Eg , if max clients is 100 (leaf max clients , range 1 .. max) then priority range is also from 1 - max) So the leaf max-clients { .. range 1- 32 } represents priority information also , so it can be max-clients or max-priorities ??

    1. (Andy)The term ’max’ in YANG resolves to 4B-1 in this range, not 32. Actually, the text I sent allows for multiple clients to all have the default priority which is not good -- if client-id is needed to resolve collisions then there is no point to requiring a unique priority per client.

    2. (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).

3. Definitions Related to Ephemeral Configuration

Currently the configuration systems managed by NETCONF ([RFC6241]) or RESTCONF ([I-D.ietf-netconf-restconf]) has 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.

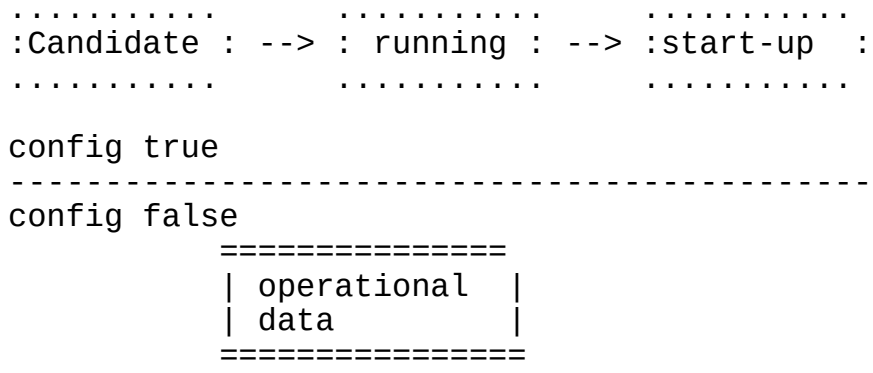

Figure 1

In reality, the running configuration becomes the intended configuration that is intended to be loaded into a device. The loading process of the intended configuation into a devices compares it against the actual devices and creates the actual configuration loaded into a box.

Some people denote the actual configuration as applied configuration. The [I-D.openconfig-netmod-opstate] denotes the actual configuration as derived state. This document will use the term actual configuration.

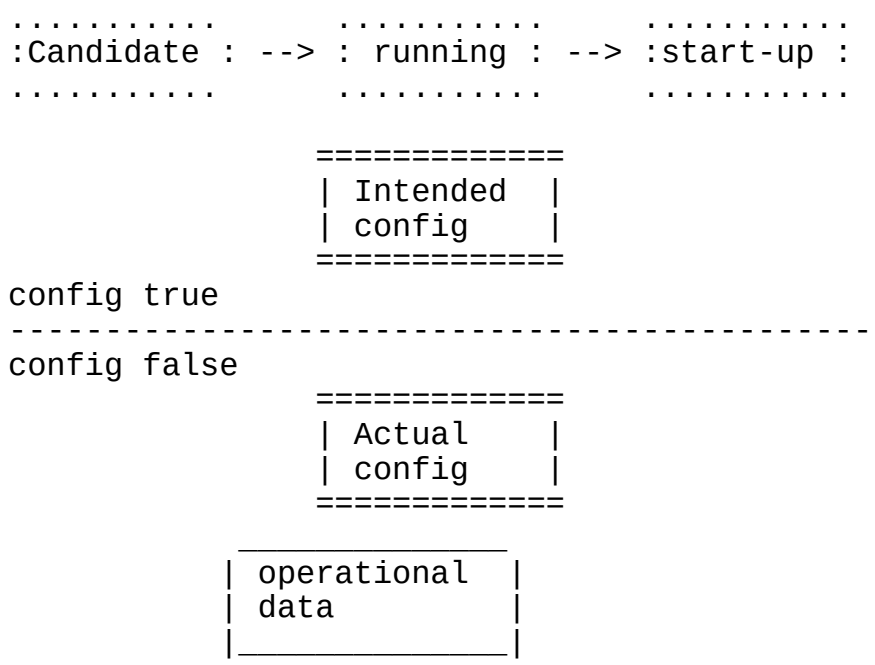

Figure 2

Recently the [I-D.openconfig-netmod-opstate] has proposed that intended configuration, actual configuration, and the traditional type of operational data included as operational state. Operational data may include:

o derived state (e.g. negotiated bgp hold timer)

o operational state for counters or statistics (interface counters)

Again, this document will use the definitions above to discuss ephemeral state until the NETCONF WG agrees upon the changes to the state diagrams.

4. 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.

The ephemeral data store has the following qualities:

1. Ephemeral state is not unique to I2RS work.

2. The ephemeral datastore is a datastore holds ephemeral configuration information that is intended to not survive a reboot. Configuration information is defined as "config=trude nodes". Ephemeral nodes will be denoted by an "ephemeral config statement in the yang protocol.

3. The ephemeral datastore is never locked.

4. The ephemeral datastore is one pane of glass that overrides the running data store.

5. Each client has a unique priority

   * client identities and priorities are assigned outside of I2RS. Examples of mechanisms are AAA or adminstrative.

   * A valid I2RS client must have both an identity and a priority.

   * 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 3

5. I2RS Error handling

Error handling is an I2RS protocol features. Normal error handling of I2RS Agent for an I2RS client’s information examines the following:

o syntax of I2RS commands,

o syntactic validation for nodes of data models,

o permission to write nodes of data model,

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

Question of editor: Should grouping be tagged somehow?

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 are allowed only for data writes which are not a part of a grouping within a data model. Groupings are defined by data models. The definition of these 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.)

RESTCONF has a complete set of operations per message. The RESTCONF patch can support accesing multiple data messages.

NETCONF stop-on-error and continue-on-error are not going to work. There is no mandated processing order for edits. For the stop-on-error and the continue-on-error process to work, the I2RS protocol extensions to NETCONF will have to force some processing order in order to support partial edits.

NETCONF has no current mechanism for reporting which edits were accepted and which edits were reject for partial operations. The I2RS protocol extensions will have to provide new error handling to the response data.

These features were removed from NETCONF (RFC 6241) because it was too complicated, and no company had implemented these features.

Interoperability issues must be considered in all three cases: a) all-or-nothing, b) stop-on-error, and c) continue-on-error.

6. Simple Thermostat Model

In this discussion of ephemeral configuration, this draft utilizes a simple thermostat model with the yang configuration found in figure 4.

```
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 4 - Simple thermostat model yabng

Figure 5 shows the diagram of the configuration state with the Simple thermostat model being attached to by an I2RS scheduler client receiving query information regarding intended configuration and actual configuration. Scheduler has a schedule set of temperatures to put in the thermostat. Actual temperature is operational state.

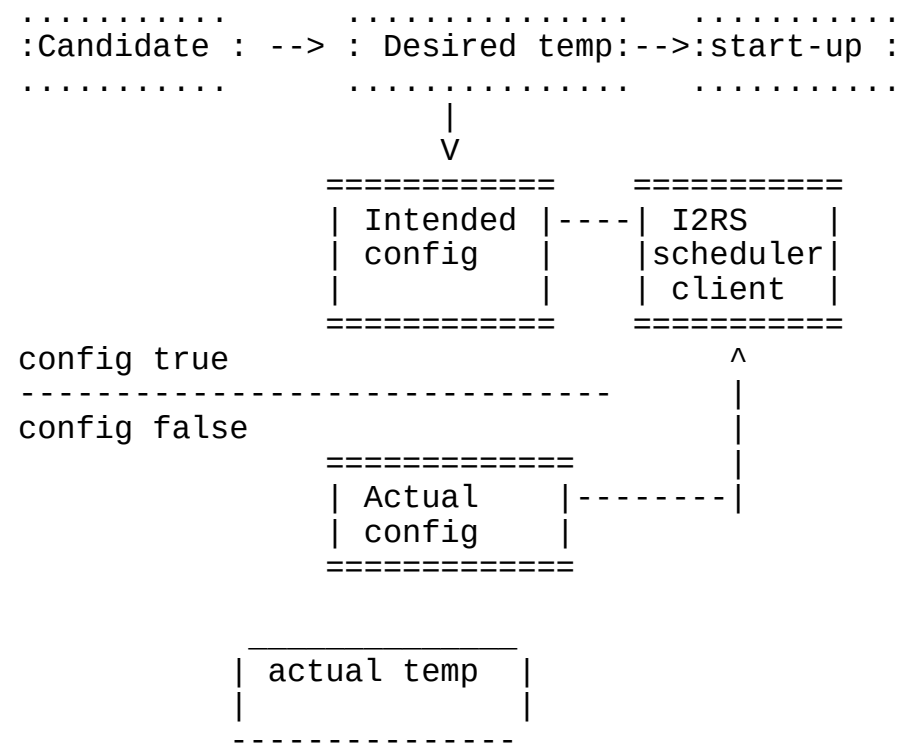

Figure 5 - Scheduler client only

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.

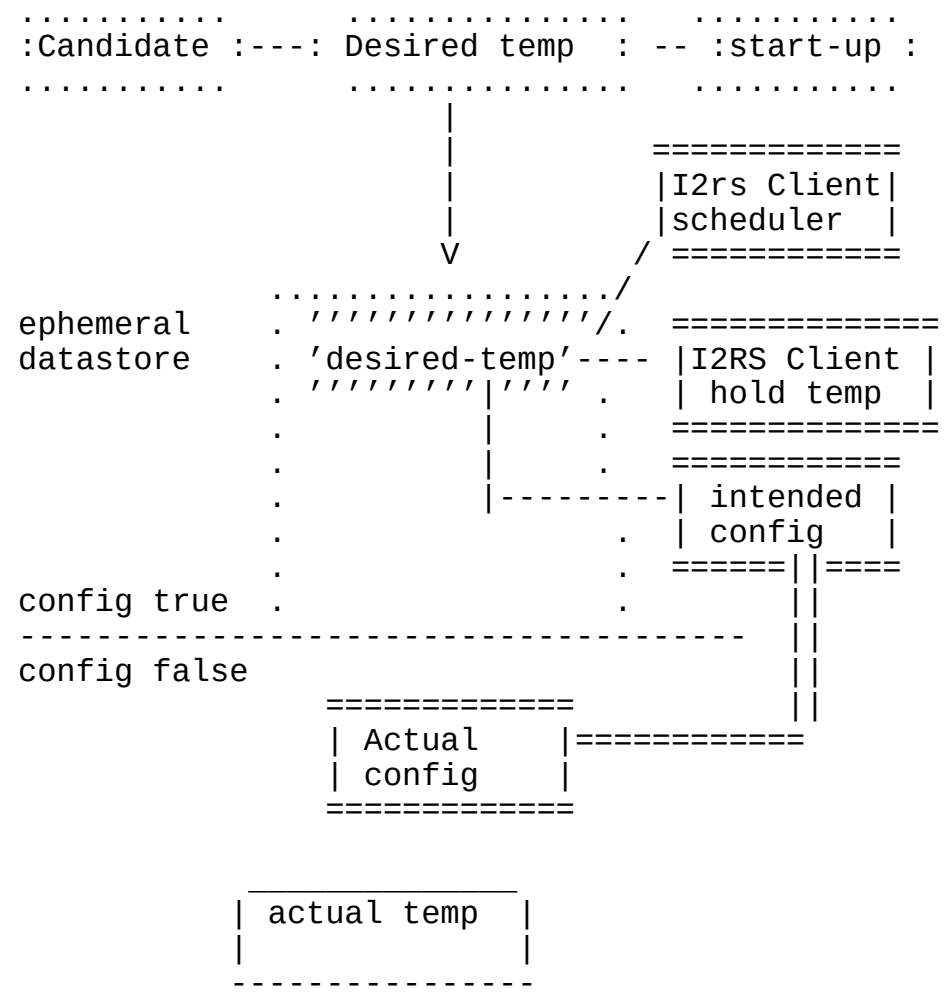

Figure 6 - Two I2RS clients

7. Yang changes

Yang needs to add a key word ephemeral that signal the ephemeral datatstore for items in the config true or the config false state.

```
module thermostat {
  ..

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

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

Figure 8 - Simple Thermostat Yang with ephemeral

Figure 6 shows the thermostat model has emphemeral variable desired-temp in the running configuration and the ephemeral data store. The RESTCONF way of addressings is below:

RESTCONF running data store

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

RESTCONF ephemeral datastore

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

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 that is intended to not survive a reboot.

- The ephemeral datastore is never locked.

- Each client has a unique priority.

- Each client writes into its own pane so there is no conflict within a pane. The implementation combines the panes into the appropriate image.

- A Partial operation is one where a subset of the written data is not applied because of better priority for that node. A partial operation is only allowed if the error-option is stop-on-error or continue-on-error.

- Caching is optional and and a server may retain the pain for each client.

8.2. Dependencies

The Yang data modules must be flag with the ephemeral data store. 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. Bulk-write

The bulk-write goes here.

8.4.2. Bulk-Read

The bulk-read goes here.

8.5. Modification to existing operations

8.5.1. PUT changes

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

8.6. Interactions with Other Capabilities

TBD

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 following features:

1. The ephemeral datastore is a datastore holds ephemeral configuration information that is intended to not survive a reboot. Configuration information is defined as "config=trude nodes". Ephemeral nodes will be denoted by an "ephemeral config statement in the yang protocol.

2. The ephemeral datastore is never locked.

3. The ephemeral datastore is one pane of glass that overrides the running data store.

4. Each I2RS client has a unique priority and a unique identity.

9.2. Dependencies

The Yang data modules must be flag with the ephemeral data store. The Yang modules must support the notification of write-conflicts. The Yang modules must support the yang-patch features as specified in [I-D.ietf-netconf-yang-patch].

9.3. Capability identifier

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

9.4. New Operations

9.4.1. Bulk-write

The bulk-write goes here.

9.4.2. Bulk-Read

The bulk-read goes here.

9.5. Modification to existing operations

9.5.1. PATCH changes

TBD

9.5.2. PUT changes

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

9.6. Interactions with Other Capabilities

TBD

10. IANA Considerations

TBD

11. Security Considerations

TBD

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
- Andy Bierman
- Alex Clemm
- Eric Voit
- Kent Watsen
- Jeff Haas
- Keyur Patel
- Hari
- Dean Bogdanavich

- Anu Nair
- Juergen Schoenwaelder
- Kent Watsen

13. References

13.1. Normative References:

[I-D.hares-i2rs-auth-trans]
Hares, S., Migault, D., and J. Halpern, "I2RS Security Related Requirements", draft-hares-i2rs-auth-trans-05 (work in progress), August 2015.

[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-09 (work in progress), March 2015.

[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-03 (work in progress), October 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-07 (work in progress), September 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-03 (work in progress), May 2015.

[I-D.ietf-netconf-yang-patch]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Patch Media Type", draft-ietf-netconf-yang-patch-05 (work in progress), July 2015.

[I-D.ietf-netmod-yang-metadata]
Lhotka, L., "Defining and Using Metadata with YANG", draft-ietf-netmod-yang-metadata-02 (work in progress), September 2015.

[I-D.openconfig-netmod-opstate]
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-01 (work in progress), July 2015.

13.2. Informative References

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-07 (work in progress), July 2015.

[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>.

Authors' Addresses

Susan Hares
Huawei
Saline
US

Email: shares@ndzh.com

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Kent Watsen
Juniper

Email: kwatsen@juniper.net