I2RS working group S. Hares

Internet-Draft Huawei

Intended status: Standards Track A. Beirman

Expires: April 21, 2016 YumaWorks

 K. Watsen

 Juniper

 October 19, 2015

 I2RS Protocol

 draft-hares-i2rs-protocol-00.txt

Abstract

 This document provides a 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.

 This proposal is not complete, but a start toward the I2RS work.

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 21, 2016.

Copyright Notice

 Copyright (c) 2015 IETF Trust and the persons identified as the

 document authors. All rights reserved.

 This document is subject to BCP 78 and the IETF Trust's Legal

 Provisions Relating to IETF Documents

 (http://trustee.ietf.org/license-info) in effect on the date of

 publication of this document. Please review these documents

 carefully, as they describe your rights and restrictions with respect

Hares, et al. Expires April 21, 2016 [Page 1]

Internet-Draft I2RS Protocol October 2015

 to this document. Code Components extracted from this document must

 include Simplified BSD License text as described in Section 4.e of

 the Trust Legal Provisions and are provided without warranty as

 described in the Simplified BSD License.

Table of Contents

 1. Introduction . . . . . . . . . . . . . . . . . . . . . . . . 3

 2. Definitions Related to Ephemeral Configuration . . . . . . . 4

 3. Definition of ephemeral datastore for NETCONF/RESTCONF . . . 5

 4. Error handling . . . . . . . . . . . . . . . . . . . . . . . 7

 4.1. syntax validation . . . . . . . . . . . . . . . . . . . . 7

 4.2. Referential validation . . . . . . . . . . . . . . . . . 8

 4.3. Grouping and Error handling . . . . . . . . . . . . . . . 8

 4.3.1. NETCONF Support of Partial Writes . . . . . . . . . . 8

 4.3.2. RESTCONF Support of Partial Writes . . . . . . . . . 9

 4.3.3. Initial Support of Parital Writes . . . . . . . . . . 9

 4.4. Priority Preemption . . . . . . . . . . . . . . . . . . . 9

 5. Yang Library Use by Ephemeral . . . . . . . . . . . . . . . . 9

 6. transport protocol . . . . . . . . . . . . . . . . . . . . . 10

 6.1. Secure Protocols . . . . . . . . . . . . . . . . . . . . 10

 6.2. Insecure Protocol . . . . . . . . . . . . . . . . . . . . 10

 7. Simple Thermostat Model . . . . . . . . . . . . . . . . . . . 11

 8. Yang changes . . . . . . . . . . . . . . . . . . . . . . . . 13

 9. NETCONF protocol extensions for the ephemeral datastore . . . 15

 9.1. Overview . . . . . . . . . . . . . . . . . . . . . . . . 15

 9.2. Dependencies . . . . . . . . . . . . . . . . . . . . . . 16

 9.3. Capability identifier . . . . . . . . . . . . . . . . . . 16

 9.4. New Operations . . . . . . . . . . . . . . . . . . . . . 16

 9.4.1. link-ephemeral . . . . . . . . . . . . . . . . . . . 17

 9.4.2. Bulk-write . . . . . . . . . . . . . . . . . . . . . 17

 9.4.3. Bulk-Read . . . . . . . . . . . . . . . . . . . . . . 17

 9.5. Modification to existing operations . . . . . . . . . . . 17

 9.5.1. <get-config> . . . . . . . . . . . . . . . . . . . . 17

 9.5.2. <edit-config> . . . . . . . . . . . . . . . . . . . . 18

 9.5.3. <copy-config> . . . . . . . . . . . . . . . . . . . . 19

 9.5.4. <delete-config> . . . . . . . . . . . . . . . . . . . 19

 9.5.5. <lock> and <unlock> . . . . . . . . . . . . . . . . . 19

 9.5.6. <get> . . . . . . . . . . . . . . . . . . . . . . . . 19

 9.5.7. <close-session> and <kill-session> . . . . . . . . . 19

 9.6. Interactions with Other Capabilities . . . . . . . . . . 19

 9.6.1. writable-running and candidate datastore . . . . . . 19

 9.6.2. confirmed commmit . . . . . . . . . . . . . . . . . . 20

 9.6.3. rollback-on-error . . . . . . . . . . . . . . . . . . 20

 9.6.4. validate . . . . . . . . . . . . . . . . . . . . . . 20

 9.6.5. Distinct Startup Capability . . . . . . . . . . . . . 20

 9.6.6. URL capability and XPATH capability . . . . . . . . . 20

 10. RESTCONF protocol extensions for the ephemeral datastore . . 20

Hares, et al. Expires April 21, 2016 [Page 2]

Internet-Draft I2RS Protocol October 2015

 10.1. Overview . . . . . . . . . . . . . . . . . . . . . . . . 20

 10.2. Dependencies . . . . . . . . . . . . . . . . . . . . . . 21

 10.3. Capability identifier . . . . . . . . . . . . . . . . . 21

 10.4. New Operations . . . . . . . . . . . . . . . . . . . . . 21

 10.4.1. Bulk-write . . . . . . . . . . . . . . . . . . . . . 21

 10.4.2. Bulk-Read . . . . . . . . . . . . . . . . . . . . . 21

 10.5. modification to data resources . . . . . . . . . . . . . 21

 10.6. Modification to existing operations . . . . . . . . . . 21

 10.6.1. OPTIONS changes . . . . . . . . . . . . . . . . . . 22

 10.6.2. HEAD changes . . . . . . . . . . . . . . . . . . . . 22

 10.6.3. GET changes . . . . . . . . . . . . . . . . . . . . 22

 10.6.4. POST changes . . . . . . . . . . . . . . . . . . . . 22

 10.6.5. PUT changes . . . . . . . . . . . . . . . . . . . . 22

 10.6.6. PATCH changes . . . . . . . . . . . . . . . . . . . 22

 10.6.7. DELETE changes . . . . . . . . . . . . . . . . . . . 22

 10.6.8. Query Parameters . . . . . . . . . . . . . . . . . . 23

 10.7. Interactions with Other Capabilities . . . . . . . . . . 23

 11. IANA Considerations . . . . . . . . . . . . . . . . . . . . . 23

 12. Security Considerations . . . . . . . . . . . . . . . . . . . 23

 13. Acknowledgements . . . . . . . . . . . . . . . . . . . . . . 23

 14. References . . . . . . . . . . . . . . . . . . . . . . . . . 24

 14.1. Normative References: . . . . . . . . . . . . . . . . . 24

 14.2. Informative References . . . . . . . . . . . . . . . . . 25

 Authors' Addresses . . . . . . . . . . . . . . . . . . . . . . . 26

1. Introduction

 This documents is a strawman/strawperson for the I2RS Protocol from

 early I2RS design team discussions. It focuses on the protocol

 extensions for ephemeral data store.

 This draft provides suggests the following additions to support the

 I2RS ephemeral state:

 o Yang ephemeral statement,

 o NETCONF ([RFC6241]) protocol extensions for the ephemeral data

 store,

 o 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.

Hares, et al. Expires April 21, 2016 [Page 3]

Internet-Draft I2RS Protocol October 2015

2. 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 under the

 config=true flag.

 o The candidate receives configuration changes from NETCONF

 (RESTCONF uses an abstract "unified datatstore" API.)

 o The running configuration is the configuration currently operating

 on a devices

 o 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. Currently, there is no datastore defined for

 configuration false.

 ........... ........... ...........

 :Candidate : --> : running : --> :start-up :

 ........... ........... ...........

 config true

 ---------------------------------------------

 config false

 ===============

 | operational |

 | data |

 ================

 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 configuration 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.

Hares, et al. Expires April 21, 2016 [Page 4]

Internet-Draft I2RS Protocol October 2015

 ........... ........... ...........

 :Candidate : --> : running : --> :start-up :

 ........... ........... ...........

 =============

 | Intended |

 | config |

 =============

 config true

 ---------------------------------------------

 config false

 =============

 | Actual |

 | config |

 =============

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

 | operational |

 | data |

 |\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

 Figure 2

 Recently the [I-D.openconfig-netmod-opstate] has proposed that

 intended configuration, actual configuration, and the traditional

 type of operational data be 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.

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 reconsider supporting caching.

 The ephemeral data store has the following qualities:

 1. Ephemeral state is not unique to I2RS work.

Hares, et al. Expires April 21, 2016 [Page 5]

Internet-Draft I2RS Protocol October 2015

 2. The ephemeral datastore is a datastore that holds ephemeral

 configuration information that is intended to not survive a

 reboot. Configuration information is defined as "config=true""

 nodes.

 3. Since Ephemeral is just about data not persisting over a reboot,

 so in theory every configuration data node in a YANG data model

 could be ephemeral. The importance of tagging an "ephemeral

 node" is for conformance checking. Therefore, ephemeral nodes

 need to be signaled in the conformance portions of the NETCONF

 and RESTCONF work. Conformance is signaled in the following

 ways:

 \* The conformance portion of NETCONF ([RFC6241]) is currently

 signaled 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.

 4. The ephemeral datastore is never locked.

 5. The ephemeral datastore is one pane of glass that overrides the

 running data store.

 6. Ephemeral data can be exposed in three ways:

 \* protocol YANG module with nodes that can be either non-

 ephemeral and ephemeral,

 \* protocol YANG modules with added nodes which can only be

 ephemeral, OR

 \* protocol independent YANG module which designed to be only

 ephemeral such as I2RS RIB, I2RS Topology models, and I2RS FB-

 RIB.

 However, ephemeral data nodes cannot have non-ephemeral data

 nodes within the subtree. Ephemeral sub-modules cannot have non-

 ephemeral data nodes within the module. Ephemeral modules cannot

 have non-ephemeral sub-modules or nodes within the module.

 7. Ephemeral nodes will be denoted by an "ephemeral config statement

 in the YANG protocol at the node level and at the module level.

 8. Ephemeral provides two additional error handling features:

Hares, et al. Expires April 21, 2016 [Page 6]

Internet-Draft I2RS Protocol October 2015

 1. Ephemeral data store allows for reduced error handling that may

 remove the requirements for leafref checking, MUST clauses,

 and instance identifier.

 2. Ephemeral data store allows for priority preemption of the

 write operation. Priority preemption means each I2RS client

 of the ephemeral I2RS agent (netconf server) is associated

 with a priority. Priority preemption 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 may

 provide a notification indication to entities monitoring that node.

4. Error handling

 Error handling is an I2RS protocol feature. Normal error handling of

 I2RS Agent for an I2RS client's information examines the following:

 o syntax validation for nodes of data model,

 o referential validation for nodes of data model,

 o grouping of data within a data models or across data models to

 accomplish tasks,

 o permission to write nodes of data model,

 o grouping,

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

 priority

 This section describes the ephemeral datastore's 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.

 Syntax validation of the data model included in the ephemeral data

 store should be done by the I2RS Agent (NETCONF/RESTCONF server).

Hares, et al. Expires April 21, 2016 [Page 7]

Internet-Draft I2RS Protocol October 2015

4.2. Referential validation

 The ephemeral data store normal processing may? not do the following

 referential checks: leafref, MUST, instance identifier. The removal

 of this validations allows for intelligence I2RS clients to rapidly

 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 models 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.

 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:

 o stop-on-error - means that the configuration process stops when a

 write to the configuration detects an error due to write conflict.

 o 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.

 o 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.)

4.3.1. 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.

Hares, et al. Expires April 21, 2016 [Page 8]

Internet-Draft I2RS Protocol October 2015

4.3.2. RESTCONF Support of Partial Writes

 RESTCONF has a complete set of operations per message. The RESTCONF

 patch can support accessing multiple data messages.

4.3.3. Initial Support of Partial Writes

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

 the I2RS Agent.

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 one with the highest priority is enabled to write the

 data. In the case of two clients with the sample priority attempting

 to write data, 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 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. Yang Library Use by Ephemeral

 The data modules supporting the Ephemeral datastore can use the Yang

 module to describe their datastore. Figure 4 shows the module

 library data structure. One part of the features of a module is the

Hares, et al. Expires April 21, 2016 [Page 9]

Internet-Draft I2RS Protocol October 2015

 type of ephemeral support (module level, submodule level, or node

 level with a list of nodes). A feature list gives the reference to

 the identifier for the ephemeral support. The feature references

 may allow for vendor extensions to ephemeral support for a specific

 model. Similarly, the deviation may point to a deviation for a

 ephemeral state model.

 +--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 4

6. transport protocol

6.1. 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).

6.2. Insecure Protocol

 The ephemeral database may support insecure protocols for information

 which is ephemeral state which does not engage configuration. The

 insecure protocol must be defined in conjunction with a data model or

 a subdata model.

Hares, et al. Expires April 21, 2016 [Page 10]

Internet-Draft I2RS Protocol October 2015

7. 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.

Hares, et al. Expires April 21, 2016 [Page 11]

Internet-Draft I2RS Protocol October 2015

 ........... ............... ...........

 :Candidate : --> : Desired temp:-->:start-up :

 ........... ............... ...........

 |

 V

 ============ ===========

 | Intended |----| I2RS |

 | config | |scheduler|

 | | | client |

 ============ ===========

 config true ^

 ------------------------------- |

 config false |

 ============= |

 | Actual |--------|

 | config |

 =============

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

 | actual temp |

 |(operational |

 | state |

 | (op-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.

Hares, et al. Expires April 21, 2016 [Page 12]

Internet-Draft I2RS Protocol October 2015

 ........... ............... ...........

 :Candidate :---: Desired temp : -- :start-up :

 ........... ............... ...........

 |

 | =============

 | |I2rs Client|

 | |scheduler |

 V / ============

 ................../

 ephemeral . '''''''''''''''/. ==============

 datastore . 'desired-temp'---- |I2RS Client |

 . '''''''''|'''' . | hold temp |

 . | . ==============

 . | . ============

 . |---------| intended |

 . . | config |

 . . ======||====

 config true . . ||

 -------------------------------------- ||

 config false ||

 ============= ||

 | Actual |============

 | config |

 =============

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

 | actual temp |

 | (op-state) |

 ----------------

 Figure 6 - Two I2RS clients

8. Yang changes

 Yang needs to add a key word ephemeral at the leaf node that signals

 allowing a version of desired-temp in the ephemeral datatstore in

 YANG.

Hares, et al. Expires April 21, 2016 [Page 13]

Internet-Draft I2RS Protocol October 2015

 module thermostat {

 ..

 leaf desired-temp {

 type int32;

 units "degrees Celsius";

 ephemeral true;

 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

 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

 {"desired-temp":19 }

 Figure 8 - RESTCONF setting of ephemeral state

 The NETCONF way of transmitting this data would be

Hares, et al. Expires April 21, 2016 [Page 14]

Internet-Draft I2RS Protocol October 2015

 <rpc-message-id=101

 xmlns="urn:ietf:params:xml:ns:base:1.0">

 <edit-config>

 <target>

 <ephemeral>

 </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

 figure 8 NETCONF setting of desired-temp

9. NETCONF protocol extensions for the ephemeral datastore

 capability-name: ephemeral-datastore

9.1. Overview

 This capability defines the NETCONF protocol extensions for the

 ephemeral state. The ephemeral state has the following features:

 o the ephemeral datastore is a datastore holds configuration

 information (Config=true) that is intended to not survive a

 reboot.

 o The ephemeral capability is signaled 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.

 o ephemeral data will be doted by an "ephemeral statement at the

 node, module "

 o The ephemeral datastore is never locked.

 o Each client has a unique priority.

 o 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.

Hares, et al. Expires April 21, 2016 [Page 15]

Internet-Draft I2RS Protocol October 2015

 o Ephemeral data nodes can occur as part of the following types of

 data modules:

 \* protocol dependent data models which mix non-ephemeral and

 ephemeral configuration data (config=true),

 \* protocol dependent data models which have only ephemeral data

 models,

 \* protocol independent data modules with only ephemeral data,

 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.

 o ephemeral error checking allows for two additional options:

 \* reduced error checking that remove the requirement for leafref

 checking, MUST clauses, and instance identifier validation.

 \* write operation with a priority preemption 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.

9.2. Dependencies

 The following are the dependencies for ephemeral support:

 The Yang data modules must be marked with the ephemeral flag at

 the node, sub-module and model.

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

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

9.3. Capability identifier

 The ephemeral-datastore capability is identified by the following

 capability string: (capability uri)

9.4. New Operations

Hares, et al. Expires April 21, 2016 [Page 16]

Internet-Draft I2RS Protocol October 2015

9.4.1. 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.

9.4.2. Bulk-write

 The bulk-write goes here if we need one. So far, editor cannot find

 a case.

9.4.3. Bulk-Read

 The bulk-read goes here if we need one, so far the editor cannot find

 a case.

9.5. Modification to existing operations

 The capability for :ephemeral-datastore modifies the target for

 existing operations.

9.5.1. <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.

 <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>

Hares, et al. Expires April 21, 2016 [Page 17]

Internet-Draft I2RS Protocol October 2015

9.5.2. <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 - which only replaces

 data if the existing data is owned by a lower priority client. If

 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> -sllows only <merge-priority> or <replace-

 priority>

 <error-option> - the I2RS agent agent has "stop-on-error",

 "continue-on-error", and "all-or-nothing" which follow the

 validation rules listed above. This also requires I2RS agents

 that support writes to have a "all-or-nothing"/"rollback-on-error"

 function.

 Note: The I2RS minimal function suggests that only error function

 that is required is the "all-or-nothing" function.

Hares, et al. Expires April 21, 2016 [Page 18]

Internet-Draft I2RS Protocol October 2015

 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>.

9.5.3. <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 fitlering to match the datasore.

9.5.4. <delete-config>

 The delete will delete all ephemeral nodes out of a datastore. The

 target must be changed to be ephemeral configuration and filters.

9.5.5. <lock> and <unlock>

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

 datastore.

9.5.6. <get>

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

 with the filters.

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

 The close session is modified to take a taret of "ephemeral-

 datastore" and to not release locks.

 The kill session is modified to take a target of "ephemeral-

 datastore, and to not change locks. "

9.6. Interactions with Other 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.

9.6.1. 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 - either the writable-running or

Hares, et al. Expires April 21, 2016 [Page 19]

Internet-Draft I2RS Protocol October 2015

 the candidate configuration data store. The <link-ephemeral>

 operation links the two databases.

9.6.2. confirmed commmit

 Confirmed commit capability is not supported for the ephemeral

 datastore.

9.6.3. 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), "stop-on-

 error", and "continue-on-error". The error handling with I2RS

 ephemeral state is described above. Initial implementations of the

 I2RS agent are only required to support the default "roll-back-on-

 error". The use of the rollback-on-error capability allows the

 optional support of more capabiity in enhanced I2RS nodes.

9.6.4. validate

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

 source: ephemeral-datastore

 filters: reference to data node, sub-module or module.

9.6.5. 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.

9.6.6. 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.

10. RESTCONF protocol extensions for the ephemeral datastore

 capability-name: ephemeral-datastore

10.1. 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.

Hares, et al. Expires April 21, 2016 [Page 20]

Internet-Draft I2RS Protocol October 2015

10.2. 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],

10.3. Capability identifier

 The ephemeral-datastore capability is identified by the following

 capability string: (capability uri)

10.4. New Operations

10.4.1. Bulk-write

 The bulk-write goes here.

10.4.2. Bulk-Read

 The bulk-read goes here.

10.5. 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 the I2RS pub/sub facility and the traceability functions. The

 "timestamp" with a last modified features must support the

 traceability function.

10.6. 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.

Hares, et al. Expires April 21, 2016 [Page 21]

Internet-Draft I2RS Protocol October 2015

10.6.1. 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.

10.6.2. HEAD changes

 The HEAD in retrieving the headers of a resources. It would be

 useful to change these headers to indicate the datastore a node or

 submodule or module is in.

 (editor)TBD on how HEAD can be changed to do this.

10.6.3. GET changes

 GET must be able to read from the URL and a particular datastore.

 (editor) TBD on how to filter for datastore in read.

10.6.4. POST changes

 POST must simply be able to create resources in ephemeral datastores

 and invoke operations defined in ephemeral data models using the

 rules of the ephemeral database.

10.6.5. PUT changes

 PUT must be able to reference an ephemeral module, sub-module, and

 nodes.

10.6.6. PATCH changes

 Plain PATCH must be able to update or create child resources in an

 ephemeral datastore. The PATCH for the ephemeral state must be

 changed to provide a merge or update of the original data only if the

 client is 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.

10.6.7. DELETE changes

 The phrase "?datastore=ephemeral" following an element will specify

 the ephemeral data store when deleting entry.

Hares, et al. Expires April 21, 2016 [Page 22]

Internet-Draft I2RS Protocol October 2015

10.6.8. 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 described above.

10.7. Interactions with Other Capabilities

 The ephemeral database must support subscribing to receiving

 notifications as Event stream. The ephemeral database] support in

 RESTCONF must also support passing error information regarding

 ephemeral data access over to pub/sub client and traceability client.

11. IANA Considerations

 TBD

12. Security Considerations

 TBD

13. 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

 o Alia Atlas

 o Ignas Bagdonas

 o Andy Bierman

 o Alex Clemm

 o Eric Voit

 o Kent Watsen

 o Jeff Haas

 o Keyur Patel

 o Hariharan Ananthakrishnan

 o Dean Bogdanavich

Hares, et al. Expires April 21, 2016 [Page 23]

Internet-Draft I2RS Protocol October 2015

 o Anu Nair

 o Juergen Schoenwaelder

 o Kent Watsen

14. References

14.1. 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-restconf]

 Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF

 Protocol", draft-ietf-netconf-restconf-07 (work in

 progress), July 2015.

 [I-D.ietf-netconf-yang-library]

 Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module

 Library", draft-ietf-netconf-yang-library-01 (work in

 progress), July 2015.

Hares, et al. Expires April 21, 2016 [Page 24]

Internet-Draft I2RS Protocol October 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.

 [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>.

Hares, et al. Expires April 21, 2016 [Page 25]

Internet-Draft I2RS Protocol October 2015

 [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

Hares, et al. Expires April 21, 2016 [Page 26]