

I2RS working group
Internet-Draft
Intended status: Standards Track
Expires: September 22, 2016

S. Hares
Huawei
A. Beirman
YumaWorks
A. Dass
Ericsson
March 21, 2016

I2RS protocol strawman
draft-hares-i2rs-protocol-strawman-01.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 22, 2016.

Copyright Notice

Copyright (c) 2016 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 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
1.1. Ephemeral Changes	4
1.2. Data Flow Changes	4
2. Definitions Related to Ephemeral Configuration	5
3. Definition of ephemeral datastore for NETCONF/RESTCONF	6
4. Error handling	9
4.1. Error handling: I2RS Normal handling	9
4.2. Error Handling: Multiple I2RS Clients Write Same Node	10
4.3. Error handling: Basic Impact on functions	10
4.3.1. Initial Support of Parital Writes	10
4.3.2. Future Scope of multiple messagewrites	10
4.3.3. Grouping and Error handling	11
4.4. Error Handling: Different levels of Validation (Debate topic)	11
4.4.1. Validation during security outage	12
4.4.2. Solution ideas	12
4.4.3. Impact on NETCONF/RESTCONF functions	13
5. transport protocol	15
5.1. Secure Protocols	15
5.2. Insecure Protocol	15
6. Yang Library Use by Ephemeral	16
7. Simple Thermostat Model	17
7.1. Yang changes	19
7.2. RESTCONF sequence	20
7.3. NETCONF messages	20
8. NETCONF protocol extensions for the ephemeral datastore	21
8.1. Overview	21
8.2. Dependencies	22
8.3. Capability identifier	22
8.4. New Operations	22
8.4.1. Bulk-Write	22
8.5. Modification to existing operations	23
8.5.1. <get-config>	23
8.5.2. <edit-config>	23
8.5.3. <copy-config>	24
8.5.4. <delete-config>	25
8.5.5. <lock> and <unlock>	25
8.5.6. <get>	25
8.5.7. <close-session> and <kill-session>	25

- 8.6. Interactions with Capabilities 25
 - 8.6.1. writable-running and candidate datastore 25
 - 8.6.2. confirmed commit 26
 - 8.6.3. rollback-on-error 26
 - 8.6.4. validate 26
 - 8.6.5. Distinct Startup Capability 26
 - 8.6.6. URL capability and XPATH capability 27
- 9. RESTCONF protocol extensions for the ephemeral datastore . . 27
 - 9.1. Overview 27
 - 9.2. Dependencies 27
 - 9.3. Capability identifier 27
 - 9.4. New Operations 27
 - 9.5. modification to data resources 28
 - 9.6. Modification to existing operations 28
 - 9.6.1. OPTIONS changes 28
 - 9.6.2. HEAD changes 28
 - 9.6.3. GET changes 28
 - 9.6.4. POST changes 28
 - 9.6.5. PUT changes 29
 - 9.6.6. PATCH changes 29
 - 9.6.7. DELETE changes 29
 - 9.6.8. Query Parameters 29
 - 9.7. Interactions with Notifications 29
 - 9.8. Interactions with Error Reporting 29
- 10. IANA Considerations 29
- 11. Security Considerations 30
- 12. Acknowledgements 30
- 13. Major Contributors 30
- 14. References 31
 - 14.1. Normative References: 31
 - 14.2. Informative References 33
- Authors' Addresses 33

1. Introduction

This documents is a strawman for I2RS higher level protocol. The I2RS protocol is a higher level protocol comprised of a set existing protocols which have been extended to work together to support a new interface to the routing system. Some people are suggesting only two protocols should be defined: NETCONF [RFC6241], and RESTCONF [I-D.ietf-netconf-restconf]. Others are suggesting we should include other data protocols.

This draft is input to a NETCONF review and design team. Many items have been settled on. Some items are in debate and those titles of those sections are marked.

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:

- 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

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 policy 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.

- o The candidate receives configuration changes from NETCONF/RESTCONF.
- 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. However, at this point there is no data stored defined for configuration false.

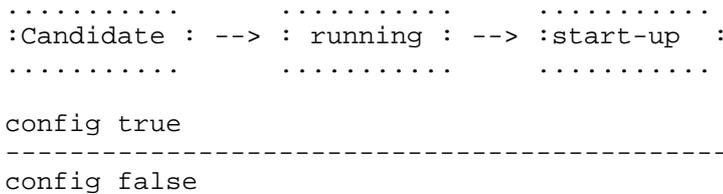


Figure 1

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.

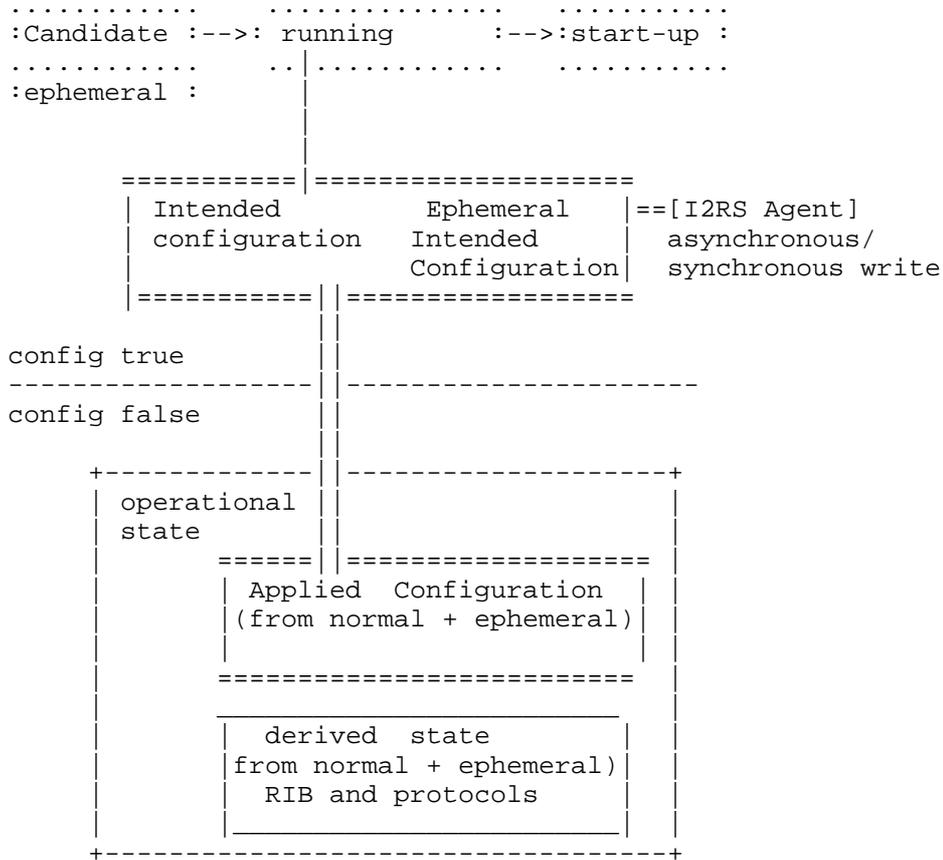


Figure 3

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 intended configuration that does not persist over a reboot.
- * Since Ephemeral is just about data not persisting 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. [I-D.hares-i2rs-bgp-dm],
 - * 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 the 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

This section will go over I2RS normal error handling, error handling when multiple I2RS clients write to the same node, and suggested alterations to the validation process for nodes.

Editor's note: The requirement for alterations to validation needs to be confirmed.

4.1. Error handling: I2RS Normal handling

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

- o message syntax validation,
- o syntax validation for nodes of data model,
- o removes referential requirements for leafref checking, MUST clauses, and instance identifier,
- o grouping of data within a data model 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

The full error handling status includes all checks included for any normal YANG data module used by NETCONF/RESTCONF. This includes

referential checks for leafref checks, MUST clauses, and instance identifiers.

If the I2RS protocol allows agents to set permissible range of error handling for writes on a data model (none, I2RS normal, full), then those stating this requirement want to be able to change this with operator-applied settings (e.g. always request full validation).

4.2. Error Handling: Multiple I2RS Clients Write Same Node

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.

4.3. Error handling: Basic Impact on functions

4.3.1. Initial Support of Parital Writes

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

4.3.1.1. 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.

4.3.1.2. RESTCONF Support of Partial Writes

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

4.3.2. 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:

- 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.)

4.3.3. Grouping and Error handling

Yang 1.0 and Yang 1.1 provide the ability to group data in groupings, leafref lists, lists, and containers. Grouping of data within a model links to data that is logically associated with one another. Data models may logical group data across models. 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.

4.4. Error Handling: Different levels of Validation (Debate topic)

The requirement for Ephemeral nodes level of validation/error handling in the I2RS protocol have been suggested to have three types of validation based on an operator-applied policy for I2RS protocol.

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

Editor's note: Andy Bierman believes that only full-validation will work. Kent Watsen suggested the "no-referential checks". Jan Medved suggested the "syntax only checks". Three excellent engineers who

are implementing I2RS suggested these three features. The editor needs aid to discuss the details of this requirement and proposal.

The first step is to see if we can confirm the requirement. After we've confirmed the requirement, the second step is to have a detailed discussion about the pro/cons of this validation. We expect to do this at IETF95.

4.4.1. Validation during security outage

[I-D.hares-i2rs-dataflow-req] indicates that higher levels of validity need to occur during security attacks. Network security controllers communicate with routing devices with network security functions such as basic firewalls in order change firewall settings during attacks. The I2NSF WG is defining communication between the network security controllers and the NSF/vNSF functions in the routers and other network devices. [I-D.hares-i2nsf-mgtflow-reqs] describes the challenges to management information flow between NSF controllers and NSF/vNSF devices operating correctly or effectively during DDoS or network security attacks.

Higher referential checks may be useful during these periods of security attacks (DDoS or others).

4.4.2. Solution ideas

This section is written to provide ideas for that discussion.

If the I2RS protocol is required to have three levels of error handling (syntax only, no-referential, full), the following are ideas for solutions:

1. only allow full validation,
2. allow a particular set of validation (syntax checks, no-referential, all-checks) per deployments of an I2RS Agent (operator-applied selection of error checking on the whole system),
3. Restrict the use of the "syntax only to operator-applied error checking" (argument: if the operator wants to shoot himself in the foot, fine). Note any module, submodule, or node that has this feature.
4. Restrict the the use of "no-referential checking to I2RS independent protocol modules, and provide error reports of referential checks,

4.4.3. Impact on NETCONF/RESTCONF functions

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

4.4.3.1. 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.

4.4.3.2. Referential validation

The ephemeral data store normal processing does not do the following referential 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.

4.4.3.3. Grouping and Error handling

Yang 1.0 and Yang 1.1 provide the ability to group data in groupings, leafref lists, lists, and containers. Adding the ephemeral data store will add these rules to references between data stores:

1. Ephemeral node can refer to config nodes, or derived state nodes (e.g. LSP),
2. config nodes cannot refer to ephemeral intended configuration nodes, and
3. derived state nodes can refer to ephemeral configuration or configuratino nodes.
4. derived state nodes are "non-persistent" and may disappear if a protocol event occurs
5. ephemeral datastore nodes are "non-presistent" and will disappear upon a reboot of the software/hardware.

Referential checks require the above rules. Not doing referential checks could cause one or more broken references to exist in the ephemeral data base. An ephemeral data bases with broken references may crash, given faulty information, or perform wrong protocol actions.

4.4.3.4. 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 the 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

4.4.3.4.1. Andy Bierman Priority Comment

(Andy)This 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 HTTOS 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.

[RFC6536] has two extensions for security. Two extensions supporting ephemeral and insecure might look like:

```
extension ephemeral {
  description "if present in a data definition statement
    then the object is considered OK for editing as ephemeral data."
}
extension non-secure-ok {
  description "if present in data definition statement
    then the object is considered OK for non-secure transport."}
```

```
To declare a local config and ephemeral edit:
leaf both {
  i2rs:ephemeral;
  type string;
  config true;
  // Yang allows leafref/XPATH to point at config=true only
}
```

```
To declare an object ephemeral edit only
leaf eph {
  i2rs:ephemeral;
  type string;
  config false;
}
```

```
To declare a non-secure leaf
leaf in-octets {
  i2rs:nonsecure-ok;
  type yang:counter64;
  config false;
}
```

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:

- o protocol version support - "version 1",
- o ephemeral model scope - ephemeral modules, mixed config module (ephemeral and config), mixed derived state (ephemeral and config).
- o multiple message support - "all or nothing",
- o pane of glass support - "single only".
- o protocol supported - "NETCONF", "RESTCONF", "NETCONF pub-sub push",
- o encoding support - XML or JSON

```

o transports protocol supported: "TCP", "SSH", "TLS", non-secure,
  and others.

o configuration for non-secure transport (An example is

  * i2rs:nonsecure-ok;

  )

+--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

Editor's Note: One feature under debate is data modules providing different levels of check on rpc or writes.

ephemeral checking - syntax only, no-referential, and full checking.

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 YANG Model

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.


```
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

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
{"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 >
    </target>
    <config>
      <top xmlns="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:

- o the ephemeral datastore is a datastore that holds configuration information (Config=true) that is intended to not survive a reboot.
- o The ephemeral capability 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.
- o ephemeral data will be noted by an "ephemeral statement at the node or module "
- o The ephemeral datastore is never locked.
- 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.

- o 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 within the module. Ephemeral modules cannot have non-ephemeral sub-modules or nodes within the module.
- o ephemeral writes have two checks: error validation and priority preemption between two I2RS client writes to the same data.
- o 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.

- o 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.

(under debate) 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. 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>
      <ephemeral-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 traceability.

<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 notification services in the pub/sub and traceability functions.

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

The existing parameters are modified as follows:

<target> - add a target of :ephemeral-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 responses 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 :ephemeral-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. I2RS ephemeral state does not impact the writeable-running data store or the candidate config datastore.

8.6.1. writable-running and candidate datastore

The writeable-running and the candidate datastore cannot be used in conjunction with the ephemeral data store. Ephemeral database overlays an intended configuration, and does not impact the writable-running or candidate data store.

8.6.2. confirmed commit

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

Editorial: Andy Bierman feels that any validation except full is going to leave the ephemeral datastore unusable. Kent Watsen suggested a "no-referential" validation as the default for I2RS protocol. Jan Medved indicated that many of the ODL Route updates are validated on the I2RS client extensively, so that the update can occur quickly with a "syntax only". Three operations people have indicated 3 different implementations. This needs to be discussed at IETF.

The text below is only a command that would provide a key word to allow three different types of validation. The command gives form to the requirements and comments from others, but it may also be broken.

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-referential - skips referential test (leafref, MUST clauses, and instance identifiers).
- * full - all normal netconf/restconf module error checking

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 datastore.

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 flagged 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],

the equivalent of the netconf pub/subscription push service found in [I-D.ietf-netconf-yang-push]

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 datastore as a context 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 strings can be done to determine the comparison of two I2RS client-priorities.

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 context ("?context=ephemeral"). Similarly, it is important the Get be able to determine if the context=ephemeral.

9.6.4. POST changes

POST must simply be able to create resources in ephemeral datastores ("context=ephemeral") and invoke operations defined in ephemeral data models.

9.6.5. PUT changes

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

9.6.6. PATCH changes

Plain PATCH must be able to update or create child resources in an ephemeral context ("?context=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 "?context=ephemeral" following an element will specify the ephemeral data store when deleting an 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 context ("?context=ephemeral") described above.

9.7. Interactions with Notifications

The ephemeral database must support the ability to publish notifications as events and the I2RS clients being able 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 must 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

- 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
- o Anu Nair
- o Juergen Schoenwaelder
- o Kent Watsen

13. Major Contributors

- o Andy Bierman (Yuman Networks) - andy@yumaworks.com
- o 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-02 (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-05 (work in progress), March 2016.
- [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-10 (work in progress), March 2016.
- [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-08 (work in progress), March 2016.
- [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.
- [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>>.
- [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

- [I-D.hares-i2nsf-mgtflow-reqs] Hares, S., "I2NSF Data Flow Requirements", draft-hares-i2nsf-mgtflow-reqs-00 (work in progress), March 2016.
- [I-D.hares-i2rs-bgp-dm] Wang, L., Hares, S., and S. Zhuang, "An I2RS BGP Data Modell", draft-hares-i2rs-bgp-dm-00 (work in progress), October 2014.
- [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>>.
- [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>>.
- [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

Amit Daas
Ericsson

Email: amit.dass@ericsson.com