I2RS working group
Internet-Draft
Intended status: Informational
Expires: September 13, 2017

S. Hares
Huawei
A. Dass
Ericsson
March 12, 2017

# NETCONF Changes to Support I2RS Protocol
# draft-hares-netconf-i2rs-netconf-01.txt

## Abstract

This document describes a NETCONF capabiilty to support the Interface to Routing system (I2RS) protocol requirements for I2RS protocol version 1. The I2RS protocol is a re-use higher layer protocol which defines extensions to other protocols (NETCONF and RESTCONf) and extensions to the Yang Data Modeling language.

The I2RS protocol supports ephemeral state datastores as control plane datastores. Initial versions of this document contain descriptions of the ephemeral datastore. Future versions may move this description to NETMOD datastore description documents.

## 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 13, 2017.

## Copyright Notice

Table of Contents

## 1. Introduction

This a proposal for Yang additions to support the first version of the I2RS protocol.

The I2RS architecture [RFC7921] defines the I2RS interface "a programmatic interface for state transfer in and out of the Internet routing system". The I2RS protocol is a protocol designed to a higher level protocol comprised of a set of existing protocols which have been extended to work together to support a new interface to the routing system. The I2RS protocol is a "reuse" management protocol which creates new management protocols by reusing existing protocols and extending these protocols for new uses, and has been designed to be implemented in phases [RFC7921].

The first version of the I2RS protocol is comprised of extensions to existing features of NETCONF [RFC6241] and RESTCONF [I-D.ietf-netconf-restconf]. The data modeling language for the I2RS protocol will be Yang [RFC7950] with features and extensions proposed in this draft.

The structure of this document is:

Section 2 provides definitions for terms in this document.

Section 3 summarizes the I2RS requirements behind these changes.

Section 4 describes the NETCONF capability to support a control protocol datastore.

Section 5 the NETCONF capability to support ephemeral state. [I-D.ietf-i2rs-ephemeral-state] specifies the I2RS requirements for the ephemeral state.

Section 6 provides a Tiny Routing Rib Yang module used by the examples in this document.

## 2. Definitions Related to Ephemeral Configuration

This section reviews definitions from I2RS architecture [RFC7921] and NETCONF operational state definitions [I-D.ietf-netmod-revised-datastores] before using these to construct a definition of the ephemeral data store.

## 2.1. Requirements language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

## 2.2. I2RS Definitions

The I2RS architecture [RFC7921] defines the following terms:

ephemeral data: is data which does not persist across a reboot (software or hardware) or a power on/off condition. Ephemeral data can be configured data or data recorded from operations of the router. Ephemeral configuration data also has the property that a system cannot roll back to a previous ephemeral configuration state. (See [RFC7921] for an architectural overview, [I-D.ietf-i2rs-ephemeral-state] for requirements, and [I-D.ietf-netmod-revised-datastores] for discussion of how the ephemeral datastore as a control plane datastore interacts with intended datstore and dynamic configuration protocols to form the applied datastore".

local configuration: is the data on a routing system which does persist across a reboot (software or hardware) and a power on/off condition. Local configuration has the ability to roll back to a pervious configuration state. Local configuration is defined as the intended datastore [I-D.ietf-netmod-revised-datastores] which is modified by dynamic configuration protocols (such as DHCP) and the I2RS ephemeral data store

dynamic configuration protocols datastore are configuration protocols such as DHCP that interact with the intended datastore (which does persist across a reboot (software or hardware) power on/off condition), and the I2RS ephemeral state control plane datastore.

operator-applied policy: is a policy that an operator sets that determines how the ephemeral datastore as a control plane data store interacts with applied datastore (as defined in [I-D.ietf-netmod-revised-datastores]). This operator policy consists of setting a priority for each of the following (per [I-D.ietf-i2rs-ephemeral-state]):

* intended configuration,

* any dynamic configuration protocols,

* any control plane datastores (one of which is ephemeral.)

# 3. Requirements control-plane datstore and ephemeral capabilities

## 3.1. I2RS protocol requirements

The requirements for the I2RS protocol are defined in the following documents:

- I2RS Problem Statement [RFC7920],
- I2RS Architecture [RFC7921],
- I2RS Traceability [RFC7922],
- Publication and Subscription [RFC7923],
- I2RS Ephemeral State Requrements, , [I-D.ietf-i2rs-ephemeral-state]
- I2RS Protocol Security Requirements, [I-D.ietf-i2rs-protocol-security-requirements]

The Interface to the routing System (I2RS) creates a new capability for the routing systems, and with greater capaiblities come a greater need for security. The requirements for a secure environment for I2RS are described in [I-D.ietf-i2rs-security-environment-reqs].

## 3.2. Overview of NETCONF support for I2RS protocol requirements

This oveview reviews the following:

- Dependencies on Existing features
- Additions to use NETCONF [RFC6241] to support control plane datastores changes get to get data, write data,(via target [merge, replace, create, delete, copy, delete-all]), close-session, kill-session, rollback-on-error (all-or-nothing), validate (validation + roll-back-on-error (all-or-nothing),
- Additions for I2RS ephemeral
- NETCONF [RFC6241] changes to obtain control-plane datastore.

# 4. NETCONF capability for control-plane datastore

capability-name: control-plane

### 4.1. Overview

This capability defines the NETCONF protocol extensions for use with control plane protocols.

A control plane datastore is not part of the configuration datastore per [I-D.ietf-netmod-revised-datastores]. The control plane datastore many contain configuration and operational state. A router implementation may merge the configuration from a control plane datastore with configuration data from the configuration datastore. A query of the applied datastore will provide a list of the installed configuration from all datastores with meta data. The current architectural provides an origin identityref with the following mappimg to datastores for the

- static - configuration data store.
- dynamic - dynamic configuration or dynamic control plane datastores.
- system - created by system,
- data-model - created by "default" in use.

Clearly, the dynamic origin title is not enough to uniquely identify a control plane datastore entry in the applied datastore. Additional definitions will need to be added to the architectural model, but this will be specified in another document.

The control plane datastores do not restrict multiple access via the locking mechanisms (<lock> and <unlock>), but use a priority scheme to handle multiple clients attempting to write the same data. The default validation within a control plane datastore’s config objects (e.g. config=TRUE) is the configuration datastore validation, but if Yang data modules specify different validation for the datastore or specific nodes then the control plane datastores will use this validation.

Some data modules may be used for both a control plane datastore and the configuration datastore. If additional validation is used for these modules, it is recommend that these modules use the "rpc" function for the additional validation rather than the <write-data> functions.

## 4.2. Dependencies

The following are the dependencies for the :control-plane capability

- Yang features:
  - [I-D.ietf-netmod-revised-datastores] functionality including the ietf-yang-architecture" data module.
  - [I-D.hares-netmod-i2rs-yang] Yang additions related to datastores definitions related to control plane datastores (datastoredef, datastore, dstype, precedence, protosup, validation), and ephemeral state.
- The following NETCONF features:
  - NETCONF [RFC6241] with its updates [RFC7803],
  - Network Access Control Model [RFC6536] with update by [I-D.ietf-netconf-rfc6536bis]
  - Running NETCONF over TLS with mutually X.509 authentication [RFC7589]
  - Keystore Model [I-D.ietf-netconf-keystore],
  - Subscribing to Yang Datastore updates [I-D.ietf-netconf-yang-push],
  - NETCONF support for Event Notifications [I-D.ietf-netconf-netconf-event-notifications],
  - Subscribing to NETCONF Events (updated) [I-D.ietf-netconf-rfc5277bis]
  - Yang Patch Media type [I-D.ietf-netconf-yang-patch],
  - NETCONF/RESTCONF Zero Touch provisioning [I-D.ietf-netconf-zerotouch],
  - TLS Client and Server Models [I-D.ietf-netconf-tls-client-server]
  - Call Home [I-D.ietf-netconf-call-home],
  - Module library [RFC7895],

* NETCONF/RESTCONF Zero Touch provisioning [I-D.ietf-netconf-zerotouch],

### 4.3. Capability identifier

The controlplane-datastore capability is identified by the following capability string: (:control-plane (uri-tbd)) where the uri-tbd is to be assigned by IANA.

### 4.4. New Operations

The following are additional protocol operations NETCONF [RFC6241] to support the following queries based on a datastore source/target datastore being specified:

- "get-data"
- "write-data"
- "validate-data"

The <target-datastore> must be registered with IANA.

#### 4.4.1. <get-data>

The get-data command has obtains configuration and operational data. The parameters the following:

source name of the datastore being queried. The valid names are "applied", "opstate", or a datstore name registered with IANA.

filter this identifies the portions of the device configuration datastore is to receive. If this parameter is not present, the entire datastore is returned. The filter MAY support subtypes "subtree", "uri", and "xpath" capabilities described in [RFC6241]. Filters may also include the elements for state (E.g. config true, config false, ephemeral true; ephemeral false;).

Positive Response If the device was able to satify the request, an <rpc-reply> is sent. The <data> section contains the appropriate subset.

Negative Response If the device was unable to satify the request, an <rpc-error> is included in the <rpc-reply>

Example - retrieve route1 in route list.
wfrom control plane datastore (cp-alpha)
and gets both configuration and ephemeral data.

```
 <rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get-data/
      <source>
        <cp-alpha/>
      </source>
      <filter type="subtree">
      <top xmlns:t=
         "http://example.com/schema/1.0/i2rs/tiny-rt-instance">
                  <route-list>
                     <route-index>1</route-index>
                  </route-list<
      </filter>
    </get-data>
 </rpc>

  <rpc-rply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get-data<
      <source>
        <cp-alpha/>
      </source>
      <filter type="subtree">
      <top xmlns:t=
         "http://example.com/schema/1.0/i2rs/tiny-rt-instance">
                  <route-list>
                     <route-index>1</route-index>
                         <prefix-match>192.2/8 /16<prefix-match>
                         <nexthop>129.1.5.1</nexthop>
                         <if-outgoing>Eth0</if-outgoing>
                         <installed>true</installed>
                  </route-list<
      </filter>
    </get-data>
 </rpc>
```

Figure 1

Example 2 - retrieve users subtree from the ephemeral database which has example control plane datastore (cp-alpha) and gets only config=true data;

```
 <rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get-data/
      <source>
        <cp-alpha/>
      </source>
      <filter type="subtree">
      <top xmlns:t=
         "http://example.com/schema/1.0/i2rs/tiny-rt-instance">
                <route-list>
                    <route-index>1</route-index>
                 </route-list<
                type="state" select "config";
      </filter>
    </get-data>
 </rpc>

  <rpc-rply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get-data<
      <source>
        <cp-alpha/>
      </source>
      <filter type="subtree">
      <top xmlns:t=
         "http://example.com/schema/1.0/i2rs/tiny-rt-instance">
                 <route-list>
                    <route-index>1</route-index>
                        <prefix-match>192.2/8 /16<prefix-match>
                        <nexthop>129.1.5.1</nexthop>
                        <if-outgoing>Eth0</if-outgoing>
                 </route-list<
      </filter>
    </get-data>
 </rpc>
```

Figure 2 - get config data

### 4.4.2. <write data gt;

The write data operationals has the attributes of operation parameters, and parameters of target datastore, default-operations, test-option, error-option, a priority, secondary-id, config, and

opstate. The attributes of the operation for individual nodes wutg "config-true" are: create, delete, merge, remove, and replace. The attributes for all-datastore operations are create-datastore, copy-datastore, delete-datastore. The operations handle multiple-client writes by using priority values rather than a locking mechanism. If two or more clients interact over changing the data node, the priority values arbitrate between the the clients where the greatest priority (1=lowest) wins. The following operations can enact changes in opstate data nodes: create, delete, remove, reset.

The default-operations parameter flags are: merge, replace, or none. The test-option parameters flags are: test-then-set, set, and test-only. The error-option oarameter flags are: stop-on-error, continue-on-error, and rollback-on-error. The priority parameter is a integer giving the priority (range 1..5000).

#### 4.4.2.1. Limitations based on other capability flags

The test-option parameters MAY only be set if the device advertises the :validate:1.1 capability. If test-option is set without the :validate:1.1 capability, an error is returned "no support for test-option".

The error-option subparameter "rollback-on-error" is enabled only if the :rollback-on-error capability is set and the data is under the config parameter.

A URL is accepted within the <source> or <target> parameters if the :url capability is set. An XPATH is accepted within the <source> or <target> parameters if the :xpath capability is set.

#### 4.4.2.2. defaults

The following are the defaults:

- The target datastore does not exists, it will be created if local support exists. Otherwise, the reply will indicate "non-supported datastore".

- If no sub-operations is specified the sub-operation, the default is merge.

- If no priority parameter is specified, the priority will be set to 1 (lowest external priority).

- If error-option is not set, then the default is "stop-on-error". (Note: for I2RS WG input. Is "stop-on-error" the same as "none"? )

- If no test-option parameter is set, the write data operates as a ’set" without validation first.

- if no secondary-identifier is given, the secondary identifier stored is "" indicating a null string.

The NETCONF priority for the "write data" function is simply the Netconf priority range. If implementations have an internal priority, the precedence between the local configuration and the NETCONF supplied configuration is set by a operator applied knob. For example, an implementation could indicate that the local configuration priority can range from 0-10 and the NETCONF priority is 10 plus the value of the priority parameter.

#### 4.4.2.3. target

The target is a datastore whose name is registered with IANA.

Example Write data

dsname = i2rs-agent

```
<rpc message-id="101"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <write data>
    <target><i2rs-agent></target>
        <operation>merge</operation>
    <priority>50<</priority>
        <error-option>all-or-nothing</error-option>
        <config>
             <top xmlns:t=
         "http://example.com/schema/1.0/i2rs/tiny-rt-instance">
                  <route-list>
                     <route-index>1</route-index>
                         <prefix-match>192.2/8 /16<prefix-match>
                         <nexthop>129.1.5.1</nexthop>
                         <if-outgoing>Eth0</if-outgoing>
                  </route-list<
              </top>
        </config>
   </write data>
```

4.4.2.4. write data operations attributes

The description of each operation is below:

<create> (config, opstate) : the creation of the data node in the datatstore if and only if the data does not already exist in the target datastore. If the data already exists, then an <rpc-error> is return wtih an error tag of "data-exists". Failure information or Success informatnoi is also pass to the notification functions (events and traceability).

<create-datastore > (config, opstate) : the creation of the datastore based on datastructures in the config and opstate parameters. The datastore ownership is set to client creating the datastore plus the priority.

<delete> (config) : the deletion of the data node if the data node exists in the configuration and either the same client deletes the node or a client with a high priority deletes the node. If configuration data does not exist in the datastore, then the <rpc-error> element is returned with a <error-tag> with value of "data-missing". The error information is passed to the notification functions to be sent as event or (optionally) placed in a tracing file.

<delete-datastore> Delete all configuration and operational data configured into the datastore, and the delete the datastore. The client requesting a delete-store must either be the owner of the datastore or have a higher priority than the client that owns the datastore. If a higher priority client takes ownership, the lower priority client is notified. If the devices is able to satisfy request, the positive response is <rcp-reply> that includes <ok> element. If the device is unable to complete request, the <rcp-reply> that includes <rpc-error> element. The operations results are forwarded to event and traceabilty functions.

<copy-datastore> If the datastore does not exist, it creates the datastore and copies the configuration values and opstate values into the datastore. The ownership information (client identity and priority) is saved as part of the datastore. If the datastore does exist and the client with ownership of the datastore changes it, then the client can replace all the datastore nodes. If a different client with lower priority than the client having ownership wants to change the datastore, the request is rejected. If a client with higher priority than the client having ownership, then the the owership changed to the new client, all the data in the datastore is deleted, all new data uploaded (config and opstate nodes). If a server is able to satisfy request, the

positive response is <rcp-reply> that includes <ok> element. If the server is unable to complete request, the <rcp-reply> that includes <rpc-error> element. The operational results are forwarded to event and traceabilty functions. If a copy-datastore action is in progress, and a client with a higher priority asks to copy-datastore, the original

<merge> (config) : parameter specifies merge. If the <priority< specifies The current data is modified by the new data in a merge of the data based on priorities. If the same client merges the data, priority is ignored. If a different client merges the data, the priority must be created than the current client's priority. If any data is replaced, this event is passed to the notification and traceability functions to pass to the setting client and the client that set the original value.

<remove> (config, opstate) : the remove of the data node if the data nodes specified in the <config> or the <opstaste> node exist. If data nodes do not exist, the "remove" operation is silently ignored and error results are forwarded to traceabilty functions.

<replace> (config) : replaces data in target if the same client replaces the top-level node, priority is ignored. If a different client replaces the note, the priority must be higher than the top level node's priority. If any data is replaced, this event is passed to the notification function (events and traceability), and a notification is sent to the previous client setting this data that the data has been reset. If the request to replace is reject due to the current top-level node having a higher priority, then an <rpc-error> returns with an error tag of "insufficient-priority". If the node is replace by a different client, the original client is notified of the change.

<reset> (opstate) : resets opstate nodes with counters to initial settings.

#### 4.4.2.5. <priority parameters>

The priority parameter sets a integer value for the priority as shown in figure x.

#### 4.4.2.6. <test-op parameter>

The <test-option> parameter performs basic function it does in the <edit-config> basic function. Just in [RFC6241], the <test-option> parameter MAY be specified only if the device advertises the ":validate:1.1" capability. The only difference is that the validation specified by the data model may augment the validation

test and the valdiation will also include the ability of the client to set this element. If a validation error occurs, the test-then-set will not perform the write-data function.

#### 4.4.2.7. <error-option> parameter

<error-option> has the following attributes

stop-on-error : Abort the write-data on the first error. This is the default.

msg-rollback-on-error : if an error condition occurs such that error serverity <rpc-error> is generated, the server will stop processing the write-data operation and restore the specific configuratoin to its complete state at the start of the "write-data" operation. This option only processes roll-back on single messages which includes

* if multiple operations occur in single message, error in one operation (E.g. read data) must not impact other operations (write-data);

* multiple operations in multiple message should be supported, but roll-back should only include a single message.

This option requires the server to support the :rollback-on-error capability.

#### 4.4.2.8. secondary-id

This operation associates a secondary identifier with a set of write-data operations. The secondary identifier is an opaque string.

## 4.5. Modification to protocol operations

### 4.5.1. Unsupported protocol operations

The following protocol operations are not supported in the control plane datastore:

- <get-config>,
- <edit-config>,
- <copy-config>,
- <delete-config>,

- o <lock>,
- o <unlock>

#### 4.5.2. Modified protocol operations

##### 4.5.2.1. <close-session> and <kill-session>

The <close-session> is modified to take a target of a control plane datastore name (registered with IANA). Since no locks are set, none should be released.

The <kill-session> is modified to take a target of a control plane datastore name (registered with IANA). Since no locks are set, none should be released.

### 4.6. Interactions with Capabilities

#### 4.6.1. Unsupported Capabiltiies

The following capabilities are not supported:

writeable-running capability,

candidate configuration capability,

confirmed commit capability,

distinct startup capbility,

#### 4.6.2. Modified Capabilities

##### 4.6.2.1. rollback-on-error

The rollback-on-error allows the error handling to be roll-back-on-error (all-or-nothing in I2RS terms) for the control plane datastore. The control plane datastore name is a valid target if the rollback-on-error capability is combined with the control plane datastore capability.

##### 4.6.2.2. validate

The validation capability engated with the control plane capability operates to validate the config portion of the control plane datastore. Therefore, the <target> is allowed to have a datstore name which is registered with IANA.

The validation of the configuration portion may contain the "validation" yang command which provides alternative validation mechanisms for specific data objects.

#### 4.6.2.3. URL capability and XPATH capability

The URL capabilities specify a <url> in the <source> and <target> operate as normal, but are allowed to specify a module within a control plane datastore.

## 5. NETCONF Ephemeral capability

capability-name: control-plane

### 5.1. Overview

The ephemeral capability is the ability to support control plane datastores which are entirely ephemeral or have ephemeral state modules, or ephemeral statements within objects in a modules. These objects can be configuration state (config=TRUE) or operational state (config=FALSE).

Ephemeral state in datastores, ephemeral modules or ephemeral objects within a module have one key characteristics: the data does not persist across reboots. The ephemeral configuration state must be restored by a client, and the operational state will need to be regenerated.

The entire requirements for ephemeral state for the I2RS control plane protocol are listed in [I-D.ietf-i2rs-ephemeral-state]. Many of these require fulfilled by the NETCONF control-plane capabilty(Ephemeral-REQ-07, Ephemeral-REQ-11, Ephemeral-REQ-12, Ephemeral-REQ-13, Ephemeral-REQ-14, Ephemeral-REQ-16).

The key features include:

- references between (to/from) ephemeral state and non-ephemeral state for constraints purposes (see Ephemeral-REQ-02, Ephemeral-REQ-03, and Ephemeral-REQ-04 in [I-D.ietf-i2rs-ephemeral-state]).

- operations to set and modify the constraints on the amount of resources the I2RS Agent (aka NETCONF server) can consume (Ephemeral-REQ-05)

- Yang modules must identify Yang objects (modules, submodules or objects within yang modules which are ephemeral and augment other nodes) and allow an "ephemeral=TRUE" feature.

## 5.2. Dependencies

Ephemeral state is not supported in the configuration datstore. The ephemeral state capability depends on having the control-plane datastore capability enabled (with appropriate NETCONF capabilities described above), and an IANA registered datastore name.

Yang must support the ability to to denote that a datastore, module, submodule or object within a module can be denoted as ephemeral. This capbility depends on the yang additions described in [I-D.hares-netmod-i2rs-yang] for control plane datastores, ephemeral key word, and validation key word.

Ephemeral state operation depends on notification of events and traceability of errors. I2RS ephemeral state requires that

## 5.3. New Operations

Note: One operation that is suggested for ephemeral state is to set resource limits. It does not seem to be an ephemeral state issue, but a control plane issue. This feature is placed here until future discussion for I2RS WG.

### 5.3.1. resource-limits

resource-limits

definition - TBD

The [I-D.ietf-i2rs-ephemeral-state] suggests setting these limits, but it does not seem to be an ephemeral function.

## 5.4. Modifications to Protocol Operations

### 5.4.1. Unsupported Operations

The ephemeral state only works as an augment to the control-plane datastore. Therefore, the following protocol operations, which are not supported in the control-plane datastore capability, are also not supported in the ephemeral capability:

- <get-config>,
- <edit-config>,
- <copy-config>,
- <delete-config>,

- o <lock>,
- o <unlock>

#### 5.4.2. Modified Operations

The ephemeral state only works as an augment to the control-plane datastore with specific ephemeral validations. Therefore, the <close-session> and <kill-session> are modified as described in the sections below.

##### 5.4.2.1. <close-session> Modifications

The <close-session> is modified to take a target of a control plane datastore name (registered with IANA). Since no locks are set, none should be released.

##### 5.4.2.2. <close-session> Modifications

The <kill-session> is modified to take a target of a control plane datastore name (registered with IANA). Since no locks are set, none should be released.

##### 5.4.2.3. <validate> Modifications

The ephemeral state may require validation to determine if the constraints obey ephemeral-state rules. If the :validate capability is used, the following parameter requires ephemeral-state contraints (Ephemeral-REQ-02, Ephemeral-REQ-03, and Ephemeral-REQ-04). If the ephemeral-constraint parameter is engaged for a module or object that is not ephemeral, the parameter is silently ignored. Error information is forwarded to the event notification processes and the traceability functions.

Additional Parameter

ephemeral-constraint

## 6. Yang model Simple Ephemeral Data model

```
datastoredef cp-alpha {
   dstype control-plane;
   description {
          "example control plane datastore ";
     }
   module-list tiny-rt-instance;
   precedence applied {
     precedenceval 5;
```

```
      }
      precedence opstate {
         precedenceval 5;
      }
   }

   datastoredef cp-beta {
      dstype control-plane i2rs-v0;
      description {
             "example control plane datastore ";
        }
      module-list tiny-rib;
      ephemeral true;
      precedence applied {
        precedenceval 50;
      }
      precedence opstate {
         precedenceval 50;
      }
   }
   module cp-example-1 {
    namespace "http://exaple.com/schema/cp-examples/1.1/tiny-rib";
    prefix trib;
    import ietf-inet-types {
       prefix inet;
    }
    import ietf-yang-types {
       prefix yang;
     }

    grouping trib-rt {
       description "tiny rib route";
       leaf route-index  {
              type uint64;
              mandatory true;
              description "route index";
            }
           leaf v4-prefix-match {
                  type inet:ipv4-prefix;
                  mandatory true;
           }
           leaf v4-nexthop {
               type inet:ipv4
                   mandatory true;
              }
           leaf if-outgoing {
               type if:interface-ref;
                   mandatory true;
```

```
                description {
                 "Name of outgoing interface";
                }
        leaf installed {
                type boolean;
        config false;
                description "rt install status ";
           }
        }
      container tiny-rt-instance {
        description
               "Tiny routing instance for
                    example purposes";
         leaf name {
            type string;
                description
                 "The name of routing instance
          which must be unique. ";
        }
         list route-list {
           key "route-index";
           description
             "a list of routes of rib"
           uses trib-rt;
         }
      }

      rpc trib-add {
       description "add route to tiny rib";
         input {
           leaf datastore {
          type string; //iana registered
          mandatory true;
          description
                    "iana datastore name";
           }
           container trib-routes {
              description
                    "Tiny rib routes to be added
                     to tiny rib";
                  list route-list {
                    key "route-index";
                    users trib-rt;
                  }
           }
        }
         output {
           container trib-add-status {
```

```
                leaf success {
                     type boolean;
                     description "add succeded";
                    }
                    leaf failure-reason {
                     type string;
                     description "reason for failure ";
                    }
               }
          }
     }
```

## 7. IANA Considerations

TBD

## 8. 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.

## 9. Acknowledgements

TBD

## 10. References

### 10.1. Normative 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>.

[RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, DOI 10.17487/RFC4107, June 2005, <http://www.rfc-editor.org/info/rfc4107>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <http://www.rfc-editor.org/info/rfc4960>.

[RFC5339] Le Roux, JL., Ed. and D. Papadimitriou, Ed., "Evaluation of Existing GMPLS Protocols against Multi-Layer and Multi-Region Networks (MLN/MRN)", RFC 5339, DOI 10.17487/RFC5339, September 2008, <http://www.rfc-editor.org/info/rfc5339>.

[RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <http://www.rfc-editor.org/info/rfc5424>.

[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>.

[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>.

[RFC6244] Shafer, P., "An Architecture for Network Management Using NETCONF and YANG", RFC 6244, DOI 10.17487/RFC6244, June 2011, <http://www.rfc-editor.org/info/rfc6244>.

[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>.

[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>.

[RFC7803] Leiba, B., "Changing the Registration Policy for the NETCONF Capability URNs Registry", BCP 203, RFC 7803, DOI 10.17487/RFC7803, February 2016, <http://www.rfc-editor.org/info/rfc7803>.

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <http://www.rfc-editor.org/info/rfc7895>.

[RFC7920] Atlas, A., Ed., Nadeau, T., Ed., and D. Ward, "Problem Statement for the Interface to the Routing System", RFC 7920, DOI 10.17487/RFC7920, June 2016, <http://www.rfc-editor.org/info/rfc7920>.

[RFC7921] Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", RFC 7921, DOI 10.17487/RFC7921, June 2016, <http://www.rfc-editor.org/info/rfc7921>.

[RFC7922] Clarke, J., Salgueiro, G., and C. Pignataro, "Interface to the Routing System (I2RS) Traceability: Framework and Information Model", RFC 7922, DOI 10.17487/RFC7922, June 2016, <http://www.rfc-editor.org/info/rfc7922>.

[RFC7923] Voit, E., Clemm, A., and A. Gonzalez Prieto, "Requirements for Subscription to YANG Datastores", RFC 7923, DOI 10.17487/RFC7923, June 2016, <http://www.rfc-editor.org/info/rfc7923>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <http://www.rfc-editor.org/info/rfc7950>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <http://www.rfc-editor.org/info/rfc7952>.

[RFC7958] Abley, J., Schlyter, J., Bailey, G., and P. Hoffman, "DNSSEC Trust Anchor Publication for the Root Zone", RFC 7958, DOI 10.17487/RFC7958, August 2016, <http://www.rfc-editor.org/info/rfc7958>.

## 10.2. Informative References

[I-D.hares-netmod-i2rs-yang]
Hares, S. and a. amit.dass@ericsson.com, "Yang for I2RS Protocol", draft-hares-netmod-i2rs-yang-04 (work in progress), March 2017.

[I-D.ietf-i2rs-ephemeral-state]
Haas, J. and S. Hares, "I2RS Ephemeral State Requirements", draft-ietf-i2rs-ephemeral-state-23 (work in progress), November 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-17 (work in progress), September 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-07 (work in progress), January 2017.

[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-10 (work in progress), December 2016.

[I-D.ietf-i2rs-security-environment-reqs]
Migault, D., Halpern, J., and S. Hares, "I2RS Environment Security Requirements", draft-ietf-i2rs-security-environment-reqs-03 (work in progress), March 2017.

[I-D.ietf-i2rs-yang-l3-topology]
Clemm, A., Medved, J., Varga, R., Liu, X., Ananthakrishnan, H., and N. Bahadur, "A YANG Data Model for Layer 3 Topologies", draft-ietf-i2rs-yang-l3-topology-08 (work in progress), January 2017.

[I-D.ietf-netconf-call-home]
Watsen, K., "NETCONF Call Home and RESTCONF Call Home", draft-ietf-netconf-call-home-17 (work in progress), December 2015.

[I-D.ietf-netconf-keystore]
Watsen, K. and G. Wu, "Keystore Model", draft-ietf-netconf-keystore-00 (work in progress), October 2016.

[I-D.ietf-netconf-netconf-event-notifications]
Prieto, A., Clemm, A., Voit, E., Nilsen-Nygaard, E., Tripathy, A., Chisholm, S., and H. Trevino, "NETCONF Support for Event Notifications", draft-ietf-netconf-netconf-event-notifications-01 (work in progress), October 2016.

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-18 (work in progress), October 2016.

[I-D.ietf-netconf-rfc5277bis]
Clemm, A., Prieto, A., Voit, E., Nilsen-Nygaard, E., Tripathy, A., Chisholm, S., and H. Trevino, "Subscribing to Event Notifications", draft-ietf-netconf-rfc5277bis-01 (work in progress), October 2016.

[I-D.ietf-netconf-rfc6536bis]
Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", draft-ietf-netconf-rfc6536bis-00 (work in progress), January 2017.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "TLS Client and Server Models", draft-ietf-netconf-tls-client-server-01 (work in progress), November 2016.

[I-D.ietf-netconf-yang-patch]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Patch Media Type", draft-ietf-netconf-yang-patch-14 (work in progress), November 2016.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscribing to YANG datastore push updates", draft-ietf-netconf-yang-push-05 (work in progress), March 2017.

[I-D.ietf-netconf-zerotouch]
Watsen, K. and M. Abrahamsson, "Zero Touch Provisioning for NETCONF or RESTCONF based Management", draft-ietf-netconf-zerotouch-12 (work in progress), January 2017.

[I-D.ietf-netmod-revised-datastores]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "A Revised Conceptual Model for YANG Datastores", draft-ietf-netmod-revised-datastores-00 (work in progress), December 2016.

[I-D.ietf-netmod-schema-mount]
Bjorklund, M. and L. Lhotka, "YANG Schema Mount", draft-ietf-netmod-schema-mount-04 (work in progress), March 2017.

[I-D.ietf-netmod-syslog-model]
Wildes, C. and K. Koushik, "A YANG Data Model for Syslog Configuration", draft-ietf-netmod-syslog-model-12 (work in progress), February 2017.

Authors' Addresses

Susan Hares
Huawei
Saline
US

Email: shares@ndzh.com

Amit Daas
Ericsson

Email: amit.dass@ericsson.com