I2RS working group
Internet-Draft
Intended status: Standards Track
Expires: December 25, 2015

J. Haas
Juniper
S. Hares
Huawei

June 23, 2015

I2RS Ephemeral State Requirements
draft-ietf-i2rs-ephemeral-state-00

Abstract

This document covers requests to the netmod and netconf Working Groups for functionality to support the ephemeral state requirements to implement the I2RS architecture.

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 December 25, 2015.

Copyright Notice

Table of Contents

1. Introduction

The Interface to the Routing System (I2RS) Working Group is chartered with providing architecture and mechanisms to inject into and retrieve information from the routing system. The I2RS Architecture document [I-D.ietf-i2rs-architecture] abstractly documents a number of requirements for implementing the I2RS requirements.

The I2RS Working Group has chosen to use the YANG data modeling language [RFC6020] as the basis to implement its mechanisms.

Additionally, the I2RS Working group has chosen to use the NETCONF [RFC6241] and its similar but lighter-weight relative RESTCONF [I-D.bierman-netconf-restconf] as the protocols for carrying I2RS.

While YANG, NETCONF and RESTCONF are a good starting basis for I2RS, there are some things needed from each of them in order for I2RS to be implemented.

2. Review of Requirements from I2RS architecture document

The following are ten requirements that [I-D.ietf-i2rs-architecture] contains which are important high level requirements:

1. The I2RS protocol SHOULD support highly reliable notifications (but not perfectly reliable notifications) from an I2RS agent to an I2RS client.

2. The I2RS protocol SHOULD support a high bandwidth, asynchronous interface, with real-time guarantees on getting data from an I2RS agent by an I2RS client.

3. The I2RS protocol will operate on data models which may be protocol independent or protocol dependent.

4. I2RS Agent needs to record the client identity when a node is created or modified. The I2RS Agent needs to be able to read the client identity of a node and use the client identity's associated priority to resolve conflicts. The secondary identity is useful for traceability and may also be recorded.

5. Client identity will have only one priority for the client identity. A collision on writes is considered an error, but priority is utilized to compare requests from two different clients in order to modify an existing node entry. Only an entry from a client which is higher priority can modify an existing entry (First entry wins). Priority only has meaning at the time of use.

6. The Agent identity and the Client identity should be passed outside of the I2RS protocol in a authentication and authorization protocol (AAA). Client priority may be passed in the AAA protocol. The values of identities are originally set by operators, and not standardized.

7. An I2RS Client and I2RS Agent mutually authenticate each other based on pre-established authenticated identities.

8. Secondary identity data is read-only meta-data that is recorded by the I2RS agent associated with a data model's node is written, updated or deleted. Just like the primary identity, the secondary identity is only recorded when the data node is written or updated or deleted

9. I2RS agent can have a lower priority I2RS client attempting to modify a higher priority client's entry in a data model. The filtering out of lower priority clients attempting to write or modify a higher priority client's entry in a data model SHOULD be effectively handled and not put an undue strain on the I2RS agent. Note: Jeff's suggests that priority is kept at the NACM at the client level (rather than the path level or the group level) will allow these lower priority clients to be filtered out using an extended NACM approach. This is only a suggestion of a method to provide the requirement 9.

10. The I2RS protocol MUST support the use of a secure transport. However, certain functions such as notifications MAY use a non-secure transport. Each model or service (notification, logging) must define within the model or service the valid uses of a non-secure transport.

3. Ephemeral State Requirements

3.1. Persistence

I2RS requires ephemeral state; i.e. state that does not persist across reboots. If state must be restored, it should be done solely by replay actions from the I2RS client via the I2RS agent.

While at first glance this may seem equivalent to the writable-running datastore in NETCONF, running-config can be copied to a persistant data store, like startup config. I2RS ephemeral state MUST NOT be persisted.

3.2. Constraints

Ephemeral state MAY refer to non-ephemeral state for purposes of implementing constraints. The designer of ephemeral state modules are advised that such constraints may impact the speed of processing ephemeral state commits and should avoid them when speed is essential.

Non-ephemeral state MUST NOT refer to ephemeral state for constraint purposes; it SHALL be considered a validation error if it does.

3.3. Hierarchy

Similar to configuration state (config true, see [RFC6020], section 7.19.1), ephemeral state is not permitted to be configured underneath nodes that are "config false" (state data).

Configuration of ephemeral state underneath "config true" is permitted. This permits augmentation of configuration state with ephemeral nodes.

Configuration of "config true" state underneath ephemeral state MUST NOT be done.

State data, "config false", is permitted underneath ephemeral state. This state data is part of the ephemeral module and should become inaccessible if the ephemeral module reboots.

4. changes to YANG

The YANG "config" keyword ([RFC6020], section 7.19.1) is extended to support the keyword "ephemeral" in addition to "true" and "false". "config ephemeral" declares the nodes underneath to be ephemeral configuration.

5. Changes to NETCONF

A capability is registered declaring that the server supports ephemeral configuration. E.g.:

```
  :ephemeral-config
       urn:ietf:params:netconf:capability:ephemeral-config:1.0
```

<get-config> will normally return "config ephemeral" nodes as it is a form of configuration. It is further extended to add a new parameter, "filter-ephemeral". This parameter accepts the following arguments:

- none (default): No filtering of persistent or ephemeral state is done.
- ephemeral-only: Only nodes representing ephemeral state are returned.
- exclude-ephemeral: Only persistent configuration is returned.

<get> is similarly extended to support "filter-ephemeral".

When a <copy-config> is done, regardless of datastore, nodes that are "config ephemeral" are excluded from the target output.

6. Requirements regarding Identity, Secondary-Identity and Priority

6.1. Identity Requirements

I2RS requires clients to have an identity. This identity will be used by the Agent authentication mechanism over the appropriate protocol.

I2RS also permits clients to have a secondary identity which may be used for troubleshooting. This secondary identity is an opaque value. [I-D.ietf-i2rs-traceability] provides an example of how the secondary identity can be used for traceability.

The secondary identity is carried in the configuration operation using a new parameter to <edit-config>. E.g.:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config>
    <i2rs:irs-secondary-identity>user1</i2rs>
    <target>
        <running/>
    </target>
    <config>
        <top xmlns="http://example.com/schema/1.2/config">
        <interface>
            <name>Ethernet0/0</name>
            <mtu>1500</mtu>
        </interface>
        </top>
    </config>
    </edit-config>
</rpc>
```

"config ephemeral" nodes that are created or altered as part of the config operation will carry the secondary-identity as read-only metadata.

6.2. Priority Requirements

To support Multi-Headed Control, I2RS requires that there be a decidable means of arbitrating the correct state of data when multiple clients attempt to manipulate the same piece of data. This is done via a priority mechanism with the highest priority winning. This priority is per-client.

This further implies that priority is an attribute that is stored in the NETCONF Access Control Model [RFC6536] as part of the group. E.g.:

```
+--rw nacm
   +--rw enable-nacm?            boolean
   +--rw read-default?           action-type
   +--rw write-default?          action-type
   +--rw exec-default?           action-type
   +--rw enable-external-groups? boolean
   +--ro denied-operations       yang:zero-based-counter32
   +--ro denied-data-writes      yang:zero-based-counter32
   +--ro denied-notifications    yang:zero-based-counter32
   +--rw groups
   |  +--rw group [name]
   |     +--rw name         group-name-type
   |     +--rw user-name*   user-name-type
   |     +--rw i2rs:i2rs-priority i2rs-priority-type
```

Ephemeral configuration state nodes that are created or altered by users that match a rule carrying i2rs-priority will have those nodes annotated with metadata. Additionally, during commit processing, if nodes are found where i2rs-priority is already present, and the priority is better than the transaction's user's priority for that node, the commit SHALL fail. An appropriate error should be returned to the user stating the nodes where the user had insufficient priority to override the state.

6.3. Representing I2RS Attributes in ephemeral configuration state

I2RS attributes may be modeled as meta-data, [I-D.ietf-netmod-yang-metadata]. This meta-data MUST be read-only; operations attempting to alter it MUST be silently ignored. An I2RS module will be defined to document this meta data. An example of its use:

```
<foo xmlns:i2rs="https://ietf.example.com/i2rs"
    i2rs:i2rs-secondary-identity="user1" i2rs:i2rs-priority="47">
  ...
</foo>
```

6.4. Semantics around storing and managing of priority and client ID.

The semantics and desired behavior around the storing and managing of priority and client ID have the following properties:

1. First - the priority mechanism is intended to handle "error cases of colliding writes" in a predictable way that results in a consistent mechanism. It is true that the same mechanism could be used if they were not considered "errors", but it is important to minimize the need and impact of the priority mechanism

2. Second, if there is a priority conflict where both clients (Client_A and Client_B) share the same priority, the client that wrote first wins. This is to avoid network oscillation if two clients are "fighting" over writing the same state. When there are multiple clients and the time arrival of the messages may not be predictable (network transit differences, which socket is read, software differences), basing state on last arrival time doesn't give consistent and predictable behavior. That gives behavior ont the following time-line

    1. Time_1: Client_A writes X=N with priority 10

    2. Time_2: Client_B attempts to write X=K with priority 10 and is rejected

    3. Time_3: Client_A writes X=P with priority 10 and succeeds

    For the I2RS Agent to properly handle these actions, it is necessary to know that X is owned by Client_A. Priority alone is not sufficient because the basis for rejecting Client_B's write but accepting Client_A's write is that Client_A is the owner. Thus it is necessary to store the Client Identity with the nodes that it owns. This could be in an I2RS-specific overlay that is only used by the I2RS agent and only contains the nodes that have been written by I2RS.

3. Third, a question has come up regarding what the behavior of priority is if a client's priority changes and whether priority needs to be stored with each node when that node is written. In my "keep-it-simple" perspective, priority is associated with a Client and is only used on a conflict. This would mean that priority is not stored with a node when that node is written. Instead, the Client Identity is stored with the node and the Client's priority is looked up in a client table that the I2RS Agent can access. That client table could be populated via configuration, via a AAA protocol, via NACM, etc. The sematic implications are as follows:

    1. Time_1: Client_A writes X=N with priority 10

    2. Time_2: Client_A's priority is changed (UNUSUAL) to priority 6

    3. Time_3: Client_B writes X=K with priority 8 (succeeds since 8 > 6)

    4. Time_4: Client_A attempts to write X=N with priority 6 (fails b/c 8 > > 6)

5. Time_5: Client_B's priority is changed (UNUSUAL) to priority 7

6. Time_6 Client_B writes X=P with priority 7 and succeeds (same > owner, no priority check)

The alternate approach would have store the priority with which a node was written. That is more like a priority lock that could only be changed by a client with higher priority or by the same client, regardless of priority. This approach would require storing a priority per node and the semantic implications would be as follows:

1. Time_1:Client_A writes X=N with priority 10

2. Time_2:Client_A's priority is changed (UNUSUAL) to priority 6

3. Time_3: Client_B attempts to write X=K with priority 8 and fails (10 > 8)

4. Time_4: Client_A writes X=N with priority 6 and succeeds (same owner, no priority check)

5. Time_5: Client_B's priority is changed (UNUSUAL) to priority 7

6. Time_6 Client_B writes X=P with priority 7 and succeeds (7 > 6)

The behavior for these two models is different at Time_3 and Time_4.

The initial preference was that the priority is not stored with the node, but if it necessary to store it with the node additional discussion may be needed with the I2RS WG.

7. Subscriptions to Changed State Requirements

I2RS clients require the ability to monitor changes to ephemeral state. While subscriptions are well defined for receiving notifications, the need to create a notification set for all ephemeral configuration state may be overly burdensome to the user.

There is thus a need for a general subscription mechanism that can provide notification of changed state, with sufficient information to permit the client to retrieve the impacted nodes. This should be doable without requiring the notifications to be created as part of every single I2RS module.

8. Transactions

Section 7.9 of the [I-D.ietf-i2rs-architecture] states the I2RS architecture does not include multi-message atomicity and rollback mechanisms, but suggests an I2RS client may inidicate one of the following error handling techniques for a given message sent to the I2RS client:

1. Perform all or none: All operations succeed or none of them will be applied. This useful when there are mutual dependencies.

2. Perform until error: Operations are applied in order, and when error occurs the processing stops. This is useful when dependencies exist between multiple-message operations, and order is important.

3. Perform all storing errors: Perform all actions storing error indications for errors. This method can be used when there are no dependencies between operations, and the client wants to sort it out.

None of these three cases insert known errors into the I2RS ephemeral datastore.

RESTCONF does an atomic action within a http session, and NETCONF has atomic actions within a commit. These features may be used to perform these features.

I2RS processing is dependent on the I2RS model. The I2RS model must consider the dependencies within multiple operations work within a model.

9. Previously Considered Ideas

9.1. A Separate Ephemeral Datastore

The primary advantage of a fully separate datastore is that the semantics of its contents are always clearly ephemeral. It also provides strong segregation of I2RS configuration and operational state from the rest of the system within the network element.

The most obvious disadvantage of such a fully separate datastore is that interaction with the network element’s operational or configuration state becomes significantly more difficult. As an example, a BGP I2RS use case would be the dynamic instantiation of a BGP peer. While it is readily possible to re-use any defined groupings from an IETF-standardized BGP module in such an I2RS

ephemeral datastore’s modules, one cannot currently reference state from one datastore to anothe

For example, XPath queries are done in the context document of the datastore in question and thus it is impossible for an I2RS model to fulfil a "must" or "when" requirement in the BGP module in the standard data stores. To implement such a mechanism would require appropriate semantics for XPath.

9.2. Panes of Glass/Overlay

I2RS ephemeral configuration state is generally expected to be disjoint from persistent configuration. In some cases, extending persistent configuration with ephemeral attributes is expected to be useful. A case that is considered potentially useful but problematic was explored was the ability to "overlay" persistent configuration with ephemeral configuration.

In this overlay scenario, persistent configuration that was not shadowed by ephemeral configuration could be "read through".

There were two perceived disadvantages to this mechanism:

The general complexity with managing the overlay mechanism itself.

Consistency issues with validation should the ephemeral state be lost, perhaps on reboot. In such a case, the previously shadowed persistent state may no longer validate.

10. Actions Required to Implement this Draft

- Draft for adding "config ephemeral" to YANG.
- Draft defining NETCONF changes including capability, RPC operation changes and support of secondary identity, RPC changes to support priority.
- I2RS draft to define meta-data for priority and secondary-identity.

11. IANA Considerations

TBD.

12. Security Considerations

TBD.

13. Acknowledgements

This document is an attempt to distill lengthy conversations on the I2RS mailing list for an architecture that was for a long period of time a moving target. Some individuals in particular warrant specific mention for their extensive help in providing the basis for this document:

- Alia Atlas
- Andy Bierman
- Martin Bjorklund
- Dean Bogdanavich
- Rex Fernando
- Joel Halpern
- Thomas Nadeau
- Juergen Schoenwaelder
- Kent Watsen

14. References

14.1. Normative References:

[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-rib-info-model]
Bahadur, N., Folkes, R., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-06 (work in progress), March 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-netmod-yang-metadata]
Lhotka, L., "Defining and Using Metadata with YANG", draft-ietf-netmod-yang-metadata-01 (work in progress), June 2015.

14.2. Informative References

[I-D.bierman-netconf-restconf]
Bierman, A., Bjorklund, M., Watsen, K., and R. Fernando, "RESTCONF Protocol", draft-bierman-netconf-restconf-04 (work in progress), February 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

Authors' Addresses

Jeff Haas
Juniper

Email: jhaas@juniper.net

Susan Hares
Huawei
Saline
US

Email: shares@ndzh.com