I2RS working group J. Haas
Internet-Draft Juniper
Intended status: Standards Track S. Hares
Expires: September 10, 2016 Huawei
March 9, 2016

I2RS Ephemeral State Requirements
draft-ietf-i2rs-ephemeral-state-04

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.

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.ietf-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

Ephemeral-REQ-01: 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 persistent data store, like startup config. I2RS ephemeral state MUST NOT be persisted.

3.2. Constraints

Ephemeral-REQ-02: Non-ephemeral state MUST NOT refer to ephemeral state for constraint purposes; it SHALL be considered a validation error if it does.

Ephemeral-REQ-03: Ephemeral state must be able to utilized temporary operational state which (MPLS LSP-ID or a BGP IN-RIB) as a constraints.

Ephemeral-REQ-04: 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.

3.3. Hierarchy

Ephemeral-REQ-05: The ability to add on an object (or a hierarchy of objects) that have the property of being ephemeral. An object needs to be able to have (both) the property of being writable and the property of the data being ephemeral (or non-ephemeral).

3.4. changes to YANG

Ephemeral-REQ-06: Yang MUST have a way to indicate in a data model that nodes have the following properties: ephemeral, writable/not-writable, status/configuration, and secure/non-secure transport.

3.5. Minimal sub-set of Changes to NETCONF

Ephemeral-REQ-07: The minimal set of changes are: (TBD).

Potential set: TBD

Note: I2RS protocol design team is working to complete this set of minimal changes.

3.6. Requirements regarding Identity, Secondary-Identity and Priority

3.6.1. Identity Requirements

Ephemeral-REQ-08:Clients shall have identifiers, and secondary identifiers.

Explanation:

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

The Secondary identities can be carried as part of RPC or meta-data. The primary purpose of the secondary identity is for traceability information which logs (who modifies certain nodes). 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.

3.6.2. 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.

Ephemeral-REQ-09: The data nodes MAY store I2RS client identity and not the effective priority at the time the data node is stored. The I2RS Client MUST have one priority at a time. The priority MAY be dynamically changed by AAA, but the exact actions are part of the protocol definition as long as Collisions are handled as described in Ephemeral-REQ-10, Ephemeral-REQ-11, and Ephemeral-REQ-12.

Ephemeral-REQ-10: When a collision occurs as two clients are trying to write the same data node, this collision is considered an error and priorities were created to give a deterministic result. When there is a collision, a notification MUST BE sent to the original client to give the original client a chance to deal with the issues surrounding the collision. The original client may need to fix their state.

Ephemeral-REQ-11: The requirement to support multi-headed control is required for collisions and the priority resolution of collisions. Multi-headed control is not tied to ephemeral state. I2RS is not mandating how AAA supports priority. Mechanisms which prevent collisions of two clients trying the same node of data are the focus.

Ephemeral-REQ-12: If two clients have the same priority, the architecture says the first one wins. The I2RS protocol has this requirement to prevent was the oscillation between clients. If one uses the last wins scenario, you may oscillate. That was our opinion, but a design which prevents oscillation is the key point.

Hints for Implementation

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 should fail. An appropriate error should be returned to the user stating the nodes where the user had insufficient priority to override the state.

3.6.3. Transactions

Ephemeral-REQ-13: Section 7.9 of the [I-D.ietf-i2rs-architecture] states the I2RS architecture does not include multi-message atomicity and roll-back mechanisms. I2RS notes multiple operations in one or more messages handling can handle errors within the set of operations

in many ways. No multi-message commands SHOULD cause errors to be inserted into the I2RS ephemeral data-store.

Explanation:

I2RS suggests the following are some of the potential error handling techniques for multiple 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.

Is important to reliability of the datastore that none of these error handling for multiple operations in one more multiple messages cause errors into be insert the I2RS ephemeral data-store.

Discussion of Current NETCONF/RESTCONF versus

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.

3.6.4. 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.

The following requirements from the [I-D.ietf-i2rs-pub-sub-requirements] apply to ephemeral state:

- o PubSub-REQ-1: The I2RS interface SHOULD support user subscriptions to data with the following parameters: push of data synchronously or asynchronously via registered subscriptions.

- o PubSSub-REQ-2: Real time for notifications SHOULD be defined by the data models.

- o PubSub-REQ-3: Security of the pub/sub data stream SHOULD be able to be model dependent.

- o PubSub-REQ-4: The Pub/Sub mechanism SHOULD allow subscription to critical Node Events. Examples of critical node events are BGP peers down or ISIS protocol overload bits.

- o PubSub-REQ-5:I2RS telemetry data for certain protocols (E.g. BGP) will require a hierarchy of filters or XPATHs. The I2RS protocol design MUST balance security against the throughput of the telemetry data.

- o PubSub-REQ-6: I2RS Filters SHOULD be able to be dynamic.

- o Pub-Sub-REQ-7: I2rs protocol MUST be able to allow I2RS agent to set limits on the data models it will support for pub/sub and within data models to support knobs for maximum frequency or resolution of pub/sub data.

4. Previously Considered Ideas

4.1. 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.

4.2. 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.

5. IANA Considerations

There are no IANA requirements for this document.

6. Security Considerations

The security requirements for the I2RS protocol are covered in [I-D.hares-i2rs-auth-trans] document.

7. 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

- o Dean Bogdanavich
- o Rex Fernando
- o Joel Halpern
- o Thomas Nadeau
- o Juergen Schoenwaelder
- o Kent Watsen

8. References

8.1. Normative References:

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

[I-D.ietf-i2rs-architecture]
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-13 (work in progress), February 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-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-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-netmod-yang-metadata]
Lhotka, L., "Defining and Using Metadata with YANG", draft-ietf-netmod-yang-metadata-04 (work in progress), February 2016.

8.2. Informative References

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-09 (work in progress), December 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

Authors' Addresses

Jeff Haas
Juniper

Email: jhaas@juniper.net

Susan Hares
Huawei
Saline
US

Email: shares@ndzh.com