Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: March 16, 2015

J. Haas, Ed.
Juniper Networks

September 12, 2014

I2RS requirements for netmod/netconf
draft-haas-i2rs-netmod-netconf-requirements-00

Abstract

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

Status of This Memo

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.

Note that this draft does not attempt to address specific implementation of I2RS requirements that the existing YANG, RESTCONF and NETCONF mechanisms are expected to cover. A separate draft will be issued for the consumption of the I2RS Working Group for such cases.

2. I2RS Requirements

2.1. Data Store Requirements

One of the key mechanisms in I2RS is the ability to inject configuration state into a network element on an ephemeral basis. 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. The author wishes to prevent any action that would lead to preserving any configuration state entered via the I2RS agent across reboots. If state has to be restored, it should be solely by replay actions from I2RS client via I2RS agent.

A few options for implementing such ephemeral configuration suggest themselves, as do some possible problems with such an implementation:

1. A separate ephemeral datastore. The semantics of this datastore is that all configuration state is known ahead of time to not survive reboot and is not to be copied into persistent storage. Such a datastore could be referenced by NETCONF and RESTCONF using existing semantics, such as "target" and "source".

2. Configuration state in the existing running datastore where the module is "tagged ephemeral".

3. Permitting existing configuration to be optionally configured as ephemeral. As an example, the NETCONF server advertises in its <hello> message if it supports the specified YANG module persistently and/or ephemerally.

2.1.1. 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 another.

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.

2.1.2. Tagged Ephemeral Modules in the Running Datastore

Presume a YANG keyword that flagged an entire module as being ephemeral. In such a case, entire modules could be crafted for I2RS (and other) purposes wherein the configuration state in the module had ephemeral properties. The primary property is that copy operations would not be able to cause the I2RS state to persist.

An obvious issue with this is the muddying of the semantics of existing NETCONF/RESTCONF operations. For example, get-config is expected to return the configuration state for the network element, but the knowledge that the configuration state may not persist is important. This may require alterations to get-config (and similar commands) along with the ambiguity of copy-config not picking up the ephemeral modules.

Providing additional parameters to the various configuration related operations in NETCONF/RESTCONF would likely be required.

2.1.3. Permitting Existing Configuration State to be Made Optionally Ephemeral

In YANG, configuration state is distinguished from operational state using "config true" vs. "config false". One way to implement I2RS state would be to introduce a third option, "config ephemeral", to configuration.

A form of this option was previously discussed in [I-D.rfernando-i2rs-yang-mods]. The suggestion of "config ephemeral" is made instead due to potential non-I2RS interest in this feature at the microphone during the IETF-90 session of netmod in Toronto, Canada.

2.2. Mutual Authentication Requirements

> "Mutual authentication between the I2RS Client and I2RS Agent is required. An I2RS Client must be able to trust that the I2RS Agent is attached to the relevant Routing Element so that write/ modify operations are correctly applied and so that information received from the I2RS Agent can be trusted by the I2RS Client."

Implementing the mutual authentication requirements for I2RS in each of the underlying protocols and their transports have some implications to be discussed.

2.2.1. NETCONF over SSH

The SSH transport does not mandate authentication be done; it is an optional feature. In an I2RS context, authentication is mandatory. Authentication of the client to the agent is carried out using any method besides "none". Authentication of the agent to the client requires that the server's public key be recognized.

2.2.2. NETCONF/RESTCONF over TLS

Agent validation of the I2RS client is mandated over TLS in an I2RS context. The client shall also validate the Agent using its server certificate.

2.3. Identity, Secondary-Identity Requirements; Priority Requirements; Implications

2.3.1. 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.

2.3.2. 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 may vary on a per-node or subtree basis based for a given identity.

2.3.3. Implications of Idenities and Priorities on Internal State

Given the requirements for I2RS identities and priority arbitration, I2RS configured state must have "meta-data" that includes the identity that caused it to come into being. Agents must also be able to map priority on a particular piece of configuration state vs. the identity provisioning it for arbitration purposes. Such mapping might be represented as part of the "meta-data" or potentially a distinct mapping database of identity vs. priority vs. configuration state. Such a mapping may be implemented using an extension to the NETCONF Access Control Model [RFC6536].

2.4. Access Control Model Requirements

2.4.1. Data Store Implications

As noted above, one of the possible options for implementing the I2RS ephemeral behavior is a separate data store. However, this clashes with Section 3.2 of [RFC6536] which limits itself to the well- known data stores.

2.4.2. I2RS Priority

A likely implementation of priority arbitration would be to extend the NACM model to also contain criteria for I2RS priority.

2.5. Connectivity Requirements

I2RS does not require clients to maintain active communication channels with their agents. Agents thus require the ability to open communication channels back to clients to satisfy previously requested information.

[I-D.ietf-netconf-call-home] describes a mechanism by which NETCONF may "phone home" using SSH and TLS.

While NETCONF notifications currently permit a different client to establish a session to an agent specifically for notification purposes, the I2RS use case typically expects that provisioning of

notifications is centrally managed and that systems receiving the notifications should not need to be individually to be provisioned.

2.6. Notification and Subscription Requirements

2.6.1. Persistence of Subscriptions

NETCONF Event Notifications [RFC5277] provides a mechanism by which subscriptions may be created. In that RFC, subscriptions are terminated when the underlying transport session ends.

As noted in the prior section, I2RS does not require its clients to maintain a connection to its agents. Thus, a mechanism by which subscriptions may be created and persist through the termination of the transport session is required.

Management of these subscriptions implies the ability to:

- Create a "headless" subscription and determine what its endpoint is.
- Specify the type and various parameters for the endpoint session. For example, the need for confidentiality may not be required for some notifications or a highly efficient transport may be required. Experience over the years has shown that very light-weight transports, for example UDP for SNMP NOTIFICATIONs/TRAPs, enables low-end components to participate in notification in a distributed fashion. A long-term implication may be that additional lighter-weight transports may be needed for I2RS notification channels.
- Delete such "headless" subscriptions.
- Enumerate the set of those subscriptions on the network element.
- Apply Access Control to the above abilities.

2.6.2. Filtering Considerations

2.6.2.1. Expressivity of Existing Filtering Mechanisms

XPath is provided as the mechanism in [RFC6241] for extended filtering. While XPath is a highly expressive language, it tends to be best suited for string and simple math-based filtering.

I2RS, being an interface to network elements, will typically have as common types IPv4 and IPv6 addresses, prefixes for the same and may require network mask interactions for firewalling. These and other

examples suggest that long term extensions appropriate for efficient filtering of I2RS types may be required.

As an example, a regular expression filter that can match addresses covered by "192.0.2.128/25" would need to match on the string "192.0.2." as a prefix and all final octets with the string representations of the values greater than or equal to 128. While methodologies for building such regular expressions are well known within the networking world, it typically results in poor performance.

2.6.2.2. Filtering Workload

Filtering, additionally, is implied to be a mechanism of the "central event processor". I2RS subscriptions may be implemented on data objects wherein a small, filtered portion of a subtree is being monitored, but the magnitude of events being generated is substantial.

Given this filtering workload, implementations are suggested to "push" the filtering to relevant system components to "pre-filter" events when possible.

2.7. Transaction Requirements

Each transaction should be treated as atomic and providing full functionality. If the configuration change is not functionally complete, then the transaction should fail and be rolled back (rollback 0). Example, I2RS agents wants to configure BGP:

```
            routing-options {
                autonomous-system autonomous-system;
            }
            protocols {
                bgp {
                    group group-name {
                        peer-as autonomous-system;
                        type type;
                        neighbor address;
                    }
                }
            }
```

If a statement like neighbor address is missing or is missformated, like 300.127.5.23, configuration is not functional, transaction should fail and rollback 0 should be performed by the I2RS agent on the ephemeral config store. If the neighbor address is in the transaction, but the address is not reachable or similar, transaction

is accepted, but notification will be sent that BGP peering can't be established.

2.8. Object Relationship Requirements

XXX JMH - I've largely convinced myself that the "instance-identifier" and "require-instance" YANG keywords satisfy the properties required by I2RS. If you don't, please supply an example.

3. IANA Considerations

This document introduces no new considerations to IANA.

4. Security Considerations

TBD

5. Acknowledgements

This document is an attempt to distill length 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 speicfic mention for their extensive help in providing the basis for this document:

- Alia Atlas

- Andy Bierman

- Martin Bjorklund

- Dean Bogdanavich

- Rex Fernando

- Joel Halpern

- Susan Hares

- Thomas Nadeau

- Juergen Schoenwaelder

- Kent Watsen

6. Normative 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.

[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-05 (work in progress), July 2014.

[I-D.ietf-netconf-call-home]
Watsen, K., "NETCONF Call Home", draft-ietf-netconf-call-home-00 (work in progress), September 2014.

[I-D.rfernando-i2rs-yang-mods]
Fernando, R., pals, p., Madhayyan, M., and A. Clemm, "YANG modifications for I2RS", draft-rfernando-i2rs-yang-mods-00 (work in progress), February 2013.

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, July 2008.

[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.

Author's Address

Jeffrey Haas (editor)
Juniper Networks
1194 N. Mathida Ave.
Sunnyvale, CA 94089
US

Email: jhaas@juniper.net