Network Working Group
Internet-Draft
Intended status: Informational
Expires: February 14, 2014

A. Atlas
Juniper Networks
J. Halpern
Ericsson
S. Hares
ADARA
D. Ward
Cisco Systems
T. Nadeau
Juniper Networks

August 13, 2013

An Architecture for the Interface to the Routing System
draft-atlas-i2rs-architecture-02

Abstract

This document describes an architecture for a standard, programmatic interface for state transfer in and out of the Internet's routing system. It describes the basic architecture, the components, and their interfaces with particular focus on those to be standardized as part of I2RS.

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 February 14, 2014.

Table of Contents

1. Introduction

Routers that form the Internet's routing infrastructure maintain state at various layers of detail and function. For example, a typical router maintains a Routing Information Base (RIB), and implements routing protocols such as OSPF, ISIS, BGP to exchange protocol state and other information about the state of the network with other routers.

A router also has information that may be required for applications to understand the network, verify that programmed state is installed in the forwarding plane, measure the behavior of various flows, routes or forwarding entries, as well as understand the configured and active states of the router. Furthermore, routers are typically configured with procedural or policy-based instructions that tell them how to convert all of this information into the forwarding operations that are installed in the forwarding plane. It is also the active state information that describes the expected and observed operational behavior of the router.

This document sets out an architecture for a common, standards-based interface to this information. This Interface to the Routing System (I2RS) facilitates control and diagnosis of the RIB manager's state, as well as enabling network applications to be built on top of today's routed networks. The I2RS is a programmatic asynchronous interface for transferring state into and out of the Internet's routing system, and recognizes that the routing system and a router's OS provide useful mechanisms that applications could harness to accomplish application-level goals.

Fundamental to the I2RS are clear data models that define the semantics of the information that can be written and read. The I2RS provides a framework for registering for and requesting the appropriate information for each particular application. The I2RS provides a way for applications to customize network behavior while leveraging the existing routing system as much as desired.

The I2RS, and therefore this document, are specifically focused on an interface for routing data.

1.1. Functional Overview

There are four key aspects to the I2RS. First, the interface is a programmatic interface which needs to be asynchronous and offers fast, interactive access. Second, the I2RS gives access to information and state that is not usually configurable or modeled in

existing implementations or configuration protocols. Third, the I2RS gives applications the ability to learn additional, structured, filterable information and events from the router. Fourth, the I2RS will be data-model driven to facilitate extensibility and provide standard data-models to be used by network applications.

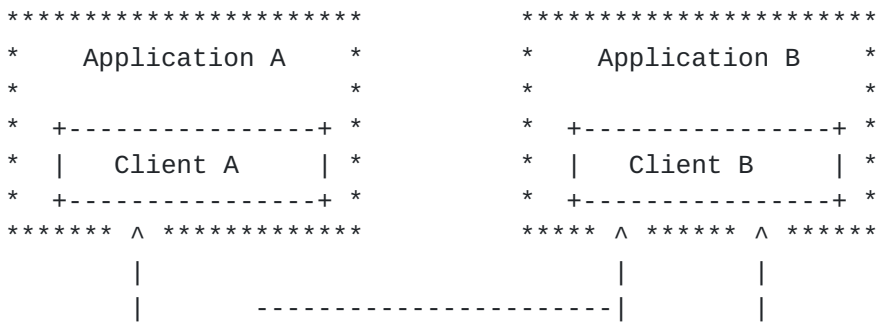
I2RS is described as an asynchronous programmatic interface; the key properties of which are described in Section 5 of [I-D.atlas-i2rs-problem-statement].

Such an interface facilitates the specification of implicitly non-permanent state into the routing system, that can optionally be made permanent. In addition, the extraction of that information and additional dynamic information from the routing system is a critical component of the interface. A non-routing protocol or application could inject state into a routing element via the state-insertion aspects of the I2RS and that state could then be distributed in a routing or signaling protocol and/or be used locally (e.g. to program the co-located forwarding plane).

There are several types of information that the I2RS will facilitate an I2RS Client obtaining. These range from dynamic event notifications (e.g. changes to a particular next-hop, interface up/down, etc.)to information collection streams (statistics, topology, route changes, etc) to simply read operations. The I2RS provides the ability for an I2RS client to request filtered and thresholded information as well as events.

1.2. Architectural Overview

The figure in Figure 1 shows the basic architecture for I2RS. Inside a Routing Element, the I2RS agent interacts with both the routing subsystem and with local configuration. A network application uses an I2RS client to communicate with one or more I2RS agents on their routing elements. The scope of I2RS is to define the interactions between the I2RS agent and the I2RS client and the associated proper behavior of the I2RS agent and I2RS client.



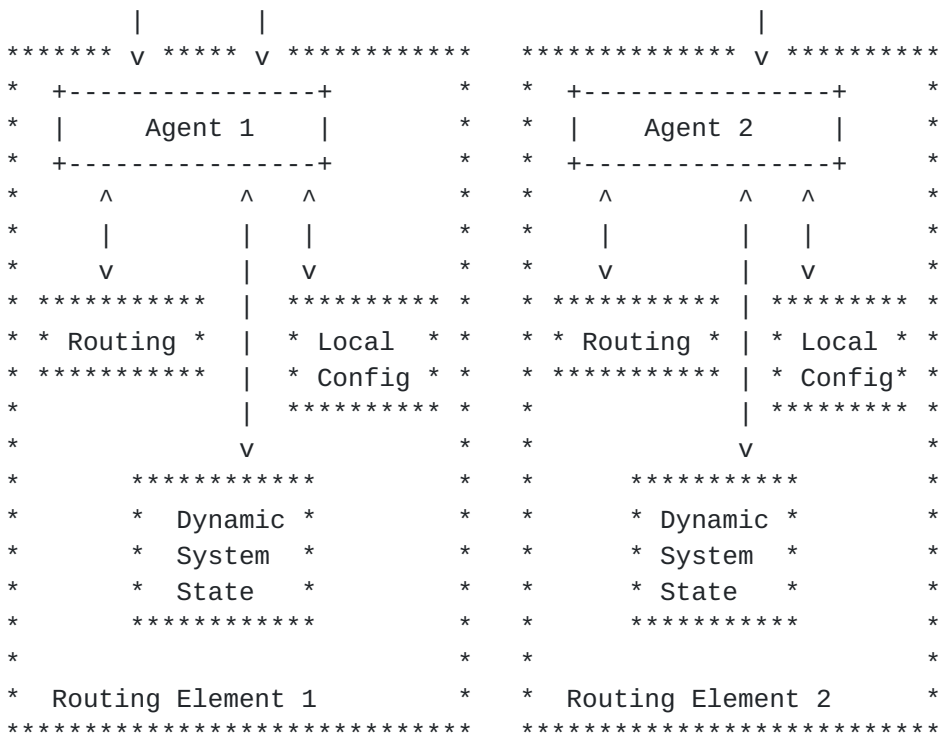


Figure 1: Architecture of I2RS clients and agents

Routing Element: A Routing Element implements at least some portion of the routing system. It does not need to have a forwarding plane associated with it. Examples of Routing Elements can include:

A router with a forwarding plane and RIB Manager that runs ISIS, OSPF, BGP, PIM, etc.

A server that runs BGP as a Route Reflector

An LSR that implements RSVP-TE, OSPF-TE, and PCEP and has a forwarding plane and associated RIB Manager.

A server that runs ISIS, OSPF, BGP and uses ForCES to control a remote forwarding plane.

A Routing Element may be locally managed, whether via CLI, SNMP, or NETCONF.

Routing: This block represents that portion of the Routing Element that implements part of the Internet routing system. It includes not merely standardized protocols (i.e. IS-IS, OSPF, BGP, PIM, RSVP-TE, LDP, etc.), but also the RIB Manager layer.

Local Config: A Routing Element will provide the ability to configure and manage it. The Local Config may be provided via a combination of CLI, NETCONF, SNMP, etc. The black box behavior for interactions between the state that I2RS installs into the routing element and the Local Config must be defined.

Dynamic System State: An I2RS agent needs access to state on a routing element beyond what is contained in the routing subsystem. Such state may include various counters, statistics, and local events. How this information is provided to the I2RS agent is out of scope, but the standardized information and data models for what is exposed are part of I2RS.

I2RS Agent: The I2RS agent implements the I2RS protocol(s) and interacts with the routing element to provide specified behavior.

Application: A network application that needs to manipulate the network to achieve its service requirements.

I2RS Client: The I2RS client implements the I2RS protocol(s). It interacts with other elements of the policy, provisioning, and configuration system by means outside of the scope of the I2RS effort. It interacts with the I2RS agents to collect information from the routing and forwarding system. Based on the information and the policy oriented interactions, the I2RS client may also interact with the I2RS agent to modify the state of the routing system the client interacts with to achieve operational goals.

As can be seen in Figure 1, an I2RS client can communicate with multiple I2RS agents. An I2RS client may connect to one or more I2RS agents based upon its needs. Similarly, an I2RS agent may communicate with multiple I2RS clients - whether to respond to their requests, to send notifications, etc. Timely notifications are critical so that several simultaneously operating applications have up-to-date information on the state of the network.

As can also be seen in Figure 1, an I2RS Agent may communicate with multiple clients. Each client may send the agent a variety of write operations. The handling of this situation has been a source of discussion in the working group. In order to keep the protocol simple, the current view is that two clients should not be attempting to write (modify) the same piece of information. Such collisions may happen, but are considered error cases that should be resolved by the network applications and management systems.

Multiple I2RS clients may need to supply data into the same list (e.g. a prefix or filter list); this is not considered an error and must be correctly handled. The nuances so that writers do not normally collide should be handled in the information models.

The architectural goal for the I2RS is that such errors should produce predictable behaviors, and be reportable to interested clients. The details of the associated policy is discussed in Section 6.8. The same policy mechanism (simple priority per I2RS client) applies to interactions between the I2RS agent and the CLI/ SNMP/NETCONF as described in Section 5.3.

In addition it must be noted that there may be indirect interactions between write operations. Detection and avoidance of such interactions is outside the scope of the I2RS work and is left to agent design and implementation for now. [[Editor's note: This topic needs more discussion in the working group.]]

2. Terminology

The following terminology is used in this document.

agent or I2RS Agent: An I2RS agent provides the supported I2RS services to the local system's routing sub-systems. The I2RS agent understands the I2RS protocol and can be contacted by I2RS clients.

client or I2RS Client: A client speaks the I2RS protocol to communicate with I2RS Agents and uses the I2RS services to accomplish a task. An I2RS client can be seen as the part of an application that uses and supports I2RS and could be a software library.

service or I2RS Service: For the purposes of I2RS, a service refers to a set of related state access functions together with the policies that control their usage. The expectation is that a service will be represented by a data-model. For instance, 'RIB service' could be an example of a service that gives access to state held in a device's RIB.

read scope: The set of information which the I2RS client is authorized to read. This access includes the permission to see the existence of data and the ability to retrieve the value of that data.

write scope: The set of field values which the I2RS client is authorized to write (i.e. add, modify or delete). This access can restrict what data can be modified or created, and what specific value sets and ranges can be installed.

scope: When unspecified as either read scope or write scope, the term scope applies to both the read scope and write scope.

resources: A resource is an I2RS-specific use of memory, storage, or execution that a client may consume due to its I2RS operations. The amount of each such resource that a client may consume in the context of a particular agent can be constrained based upon the client's security role. An example of such a resource could include the number of notifications registered for. These are not protocol-specific resources or network-specific resources.

role or security role: A security role specifies the scope, resources, priorities, etc. that a client or agent has.

identity: A client is associated with exactly one specific identity. State can be attributed to a particular identity. It is possible for multiple communication channels to use the same identity; in that case, the assumption is that the associated client is coordinating such communication.

secondary identity: An I2RS Client may supply a secondary opaque identity that is not interpreted by the I2RS Agent. An example use is when the I2RS Client is a go-between for multiple applications and it is necessary to track which application has requested a particular operation.

3. Key Architectural Properties

3.1. Simplicity

There have been many efforts over the years to improve the access to the information known to the routing and forwarding system. Making such information visible and usable to network management and applications has many well-understood benefits. There are two related challenges in doing so. First, the span of information potentially available is very large. Second, the variation both in the structure of the data and in the kinds of operations required tends to introduce protocol complexity.

Having noted that, it is also critical to the utility of I2RS that it be easily deployed and robust. Complexity in the protocol hinders implementation, robustness, and deployability. Also, complexity in the data models frequently makes it harder to extend rather than easier.

Thus, one of the key aims for I2RS is the keep the protocol and modeling architecture simple. So for each architectural component or aspect, we ask ourselves "do we need this complexity, or is the behavior merely nice to have?" Protocol parsimony is clearly a goal.

3.2. Extensibility

There are several ways that the scope of the I2RS work is being restricted in the interests of achieving a deliverable and deployable result. We are only working on the models to be used over the single identified interface. We are only looking at modeling a subset of the data of interest. And we are probably only representing a subset of the operations that may eventually be needed (although there is some hope that we are closer on that aspect than others to what is needed.) Thus, it is important to consider extensibility not only of the underlying services' data models, but also of the primitives and protocol operations.

At the same time, it is clearly desirable for the data models and protocol operations we define in the I2RS to be useful the in more general settings. It should be easy to integrate data models from the I2RS with other data. Other work should be able to easily extend it to represent additional aspects of the network elements or network systems. Hence, the data model and protocol definitions need to be designed to be highly extensible, preferably in a regular and simple fashion.

3.3. Model-Driven Programmatic Interfaces

A critical component of I2RS is the standard information and data models with their associated semantics. While many components of the routing system are standardized, associated data models for them are not yet available. Instead, each router uses different information, different mechanisms, and different CLI which makes a standard interface for use by applications extremely cumbersome to develop and maintain. Well-known data modeling languages exist and may be used for defining the data models for I2RS.

There are several key benefits for I2RS in using model-driven architecture and protocol(s). First, it allows for transferring data-models whose content is not explicitly implemented or understood. Second, tools can automate checking and manipulating

data; this is particularly valuable for both extensibility and for the ability to easily manipulate and check proprietary data-models.

The different services provided by I2RS can correspond to separate data-models. An I2RS agent may indicate which data-models are supported.

3.4. Authorization and Authentication

All control exchanges between the I2RS client and agent MUST be authenticated and integrity protected (such that the contents cannot be changed without detection). Manipulation of the system must be accurately attributable. In an ideal architecture, even information collection and notification should be protected; this may be subject to engineering tradeoffs during the design.

I2RS Agents, in performing information collection and manipulation, will be acting on behalf of the I2RS clients. As such, they will operate based on the lower of the two permissions of the agent itself and of the client.

I2RS clients may be operating on behalf of other applications. While those applications' identities are not need for authorization, each application should have a unique opaque identifier that can be provided by the I2RS client to the I2RS agent for purposes of tracking attribution of operations to support functionality such as accounting and troubleshooting.

4. Network Applications and I2RS Client

An I2RS Client has a standardized interface that uses the I2RS protocol(s) to communicate with I2RS Agents. The interface between an I2RS client and the network applications is outside the scope of I2RS.

When an I2RS Client interacts with multiple network applications, that I2RS Client is behaving as a go-between and should indicate this to the I2RS Agents by, for example, specifying a secondary opaque identity to allow improved troubleshooting.

A network application that uses an I2RS client may also be considered a routing element and include an I2RS agent for interactions. However, where the needed information and data models for that upper interface differs from that of a conventional routing element, those models are, at least initially, out of scope for I2RS.

4.1. Example Network Application: Topology Manager

One example of such an application is a Topology Manager. A Topology Manager includes an I2RS client that uses the I2RS data models and protocol to collect information about the state of the network by communicating directly with one or more I2RS agents. From these I2RS agents, the Topology Manager collects routing configuration and operational data. Most importantly, it collects information about the routing system, including the contents of the IGP (e.g., IS-IS or OSPF) and BGP data sets.

The Topology Manager may be embedded as a component of a larger application. It would construct internal data structures and use the collected data to drive functions such as path computations or anomalous routing detection. Alternatively, the Topology Manager could combine the I2RS-collected data with other information, abstract a composite set, and provide a coherent picture of the network state accessible via another interface. That interface might use the same I2RS protocol and could use extensions to the I2RS data models. Developing such mechanisms is outside the initial scope of the I2RS work.

5. I2RS Agent Role and Functionality

The I2RS Agent is part of a routing element. As such, it has relationships with that routing element as a whole, and with various components of that routing element.

5.1. Relationship to its Routing Element

A Routing Element may be implemented with a wide variety of different architectures: an integrated router, a split architecture, distributed architecture, etc. The architecture does not need to affect the general I2RS agent behavior.

For scalability and generality, the I2RS agent may be responsible for collecting and delivering large amounts of data from various parts of the routing element. Those parts may or may not actually be part of a single physical device. Thus, for scalability and robustness, it is important that the architecture allow for a distributed set of reporting components providing collected data from the I2RS agent back to the relevant I2RS clients. As currently envisioned, a given I2RS agent would have only one locus per I2RS service for manipulation of routing element state.

5.2. State Storage

State modification requests are sent to the I2RS agent in a network element by I2RS clients. The I2RS agent is responsible for applying these changes to the system. How much data must the I2RS Agent store about these state-modifying operations, and with what persistence? There are range of possible answers. One extreme is where it stores nothing, cannot indicate why or by whom state was placed into the routing element, and relies on clients reapplying things in all possible cases. The other extreme is where multiple clients' overlapping operations are stored and managed, as is done in the RIB for routes with a preference or priority to pick between the routes.

In answering this question, this architecture tries to provide sufficient power to keep client operations effective, while still being simple to implement in the I2RS Agent, and to observe meaningfully during operation. The I2RS agent stores the set of operations it has applied. Simply, the I2RS agent stores who did what operation to which entity. New changes replace any data about old ones. If an I2RS client does an operation to remove some state, that state is removed and the I2RS agent stores no more information about it. This allows any interested party to determine what the current effect of I2RS on the system is, and why. Meaningful logging is also recommended.

The I2RS Agent will not attempt to retain or reapply state across routing element reboot. Determination of whether state still applies depends heavily on the causes of reboots, and reapplication is at least as likely to cause problems as it is to provide for correct operation. [[Editor's note: This topics needs more discussion in the working group.]]

5.2.1. Starting and Ending

An I2RS client applies changes via the I2RS protocol based on policy and other application inputs. While these changes may be of the form "do this now, and leave it there forever", they are frequently driven by other conditions which may have start times, stop times, or are only to be used under certain conditions. The I2RS interface protocol could be designed to allow an I2RS Client to provide a wide range of such conditional information to the I2RS Agent for application. At the other extreme, the I2RS client could provide all such functionality based on its own clocking and network event reporting from the relevant I2RS Agents.

Given that the complexity of possible conditions is very large, and that some conditions may even cross network element boundaries, clearly some degree of handling must be provided on the I2RS client.

As such, in this architecture it is assumed that all the complexity associated with this should be left to the I2RS client. This architectural view does mean that reliability of the communication path between the I2RS client and I2RS agent is critical. [[Editor's note: This requires more discussion in the working group.]]

5.2.2. Reversion

An I2RS Agent may decide that some state should no longer be applied. An I2RS Client may instruct an Agent to remove state it has applied. In all such cases, the state will revert to what it would have been without the I2RS; that state is generally whatever was specified via the CLI, NETCONF, SNMP, etc. I2RS Agents will not store multiple alternative states, nor try to determine which one among such a plurality it should fall back to. Thus, the model followed is not like the RIB, where multiple routes are stored at different preferences.

An I2RS Client may register for notifications when state that was applied by a particular I2RS Client is modified or removed.

5.3. Interactions with Local Config

As described above, local device configuration is considered to be separate from the I2RS data store. Thus, changes may originate from either source. Policy (i.e. comparisons between a CLI/SNMP/NETCONF priority and a I2RS agent priority) can determine whether the local configuration should overwrite any state written by I2RS and attributed to a particular I2RS Client or whether I2RS as attributed to a particular I2RS Client can overwrite local configuration state.

Simply allowing the most recent state to prevail could cause race conditions where the final state is not repeatably deterministic. One important aspect is that if CLI/SNMP/NETCONF changes data that is subject to monitoring or manipulating by I2RS, then the system must be instrumented enough to provide suitable I2RS notifications of these changes.

5.4. Routing Components and Associated I2RS Services

For simplicity, each logical protocol or set of functionality that be compactly described in a separable information and data model is considered as a separate I2RS Service. A routing element need not implement all routing components described nor provide the associated I2RS services. The initial services included in the I2RS architecture are as follows.

5.4.1. Unicast and Multicast RIB and LFIB

Network elements concerned with routing IP maintain IP unicast RIBs. Similarly, there are RIBs for IP Multicast, and a Label Information Base (LIB) for MPLS. The I2RS Agent needs to be able to read and write these sets of data. The I2RS data model must include models for this information.

In particular, with regard to writing this information, the I2RS Agent should use the same mechanisms that the routing element already uses to handle RIB input from multiple sources, so as to compatibly change the system state.

The multicast state added to the multicast RIB does not need to match to well-known protocol installed state. The I2RS Agent can create arbitrary replication state in the RIB, subject to the advertised capabilities of the routing element.

5.4.2. IGPs, BGP and Multicast Protocols

In addition to interacting with the consolidated RIB, the I2RS agent may need to interact with the individual routing protocols on the device. This interaction includes a number of different kinds of operations:

- reading the various internal rib(s) of the routing protocol is often helpful for understanding the state of the network. Directly writing these protocol-specific RIBs or databases is out of scope for I2RS.

- reading the various pieces of policy information the particular protocol instance is using to drive its operations.

- writing policy information such as interface attributes that are specific to the routing protocol or BGP policy that may indirectly manipulate attributes of routes carried in BGP.

- writing routes or prefixes to be advertised via the protocol.

- joining/removing interfaces from the multicast trees

For example, the interaction with OSPF might include modifying the local routing element's link metrics, announcing a locally-attached prefix, or reading some of the OSPF link-state database. However, direct modification of of the link-state database is NOT allowed to preserve network state consistency.

5.4.3. MPLS

The I2RS agent will need to interact with the protocols that create transport LSPs (e.g. LDP and RSVP-TE) as well as protocols (e.g. BGP, LDP) that provide MPLS-based services (e.g. pseudowires, L3VPNs, L2VPNs, etc).

5.4.4. Policy and QoS Mechanisms

Many network elements have separate policy and QoS mechanisms, including knobs which affect local path computation and queue control capabilities. These capabilities vary widely across implementations, and I2RS cannot model the full range of information collection or manipulation of these attributes. A core set does need to be included in the I2RS data models and in the expected interfaces between the I2RS Agent and the network element, in order to provide basic capabilities and the hooks for future extensibility. [[Editor's note: This requires more discussion in the working group.]]

6. I2RS Client Agent Interface

6.1. Protocol Structure

One could view I2RS merely as a way to talk about the existing network management interfaces to a network element. That would be quite limiting and would not meet the requirements elucidated elsewhere. One could also view I2RS as a collection of protocols - some existing and some new - that meet the needs. While that could be made to work, the complexity of such a mechanism would be quite high. One would need to develop means to coordinate information across a set of protocols that were not designed to work together. From a deployability perspective, this would not meet the goal of simplicity. As a result, this architecture views the I2RS as an interface supporting a single control and data exchange protocol. Whether such a protocol is built upon extending existing mechanisms or requires a new mechanism requires further investigation. That protocol may use several underlying transports (TCP, SCTP, DCCP), with suitable authentication and integrity protection mechanisms. These different transports can support different types of communication (e.g. control, reading, notifications, and information collection) and different sets of data. Whatever transport is used for the data exchange, it must also support suitable congestion control mechanisms.

6.2. Channel

The uses of a single I2RS protocol does not imply that only one channel of communication is required. There may be a range of reliability requirements, and to support the scaling there may need to be channels originating from multiple sub-components of a routing element. These will all use the date exchange protocol, and establishment of additional channels for communication will be coordinated between the I2RS client and the I2RS agent.

6.3. Negotiation

Protocol support capabilities will vary across I2RS Clients and Routing Elements supporting I2RS Agents. As such, capability negotiation (such as which transports are supported beyond the minimum required to implement) will clearly be necessary. It is important that such negotiations be kept simple and robust, as such mechanisms are often a source of difficulty in implementation and deployment.

Negotiation should be broken into several aspects, such as protocol capablities and I2RS services and model types supported.

6.4. Identity and Security Role

Each I2RS Client will have a unique identity; it can also have secondary identities to be used for troubleshooting. A secondary identity is merely a unique, opaque identifier that may be helpful in troubleshooting. Via authentication and authorization mechanisms, the I2RS agent will have a specific scope for reading data, for writing data, and limitations on the resources that can be consumed. The scopes need to specify both the data and the value ranges.

6.4.1. Client Redundancy

I2RS must support client redundancy. At the simplest, this can be handled by having a primary and a backup network application that both use the same client identity and can successfully authenticate as such. Since I2RS does not require a continuous transport connection and supports multiple transport sessions, this can provide some basic redundancy. However, it does not address concerns for troubleshooting and accountability about knowing which network application is actually active. At a minimum, basic transport information about each connection and time can be logged with the identity. Further discussion is necessary to determine whether additional client identification information is necessary.[[Editor's note: This requires more discussion in the working group.]]

6.5. Connectivity

A client may or may not maintain an active communication channel with an agent. Therefore, an agent may need to open a communication channel to the client to communicate previously requested information. The lack of an active communication channel does not imply that the associated client is non-functional. When communication is required, the agent or client can open a new communication channel.

State held by an agent that is owned by a client should not be removed or cleaned up when a client is no longer communicating - even if the agent cannot successfully open a new communication channel to the client.

There are three different assumptions that can apply to handling dead clients. The first is that the network applications or management systems will detect a dead network application and either restart that network application or clean up any state left behind. The second is to allow state expiration, expressed as a policy associated with the I2RS client's role. The state expiration could occur after there has been no successful communication channel to or from the I2RS client for the policy-specified duration. The third is that the client could explicitly request state clean-up if a particular transport session is terminated.

6.6. Notifications

As with any policy system interacting with the network, the I2RS Agent needs to be able to receive notifications of changes in network state. Notifications here refers to changes which are unanticipated, represent events outside the control of the systems (such as interface failures on controlled devices), or are sufficiently sparse as to be anomalous in some fashion.

Such events may be of interest to multiple I2RS Clients controlling data handled by an I2RS Agent, and to multiple other I2RS clients which are collecting information without exerting control. The architecture therefore requires that it be practical for I2RS Clients to register for a range of notifications, and for the I2R Agents to send notifications to a number of Clients.

As the I2RS is developed, it is likely that a management information-model and data-model will be required to describe event notifications for general or I2RS errors.

For performance and scaling by the I2RS client and general information privacy, an I2RS Client needs to be able to register for just the events it is interested in. It is also possible that I2RS might might provide a stream of notifications via a publish/subscribe

mechanism that is not amenable to having the I2RS agent do the filtering.

6.7. Information collection

One of the other important aspects of the I2RS is that it is intended to simplify collecting information about the state of network elements. This includes both getting a snapshot of a large amount of data about the current state of the network element, and subscribing to a feed of the ongoing changes to the set of data or a subset thereof. This is considered architecturally separate from notifications due to the differences in information rate and total volume.

6.8. Multi-Headed Control

As was described earlier, an I2RS Agent interacts with multiple I2RS Clients who are actively controlling the network element. From an architecture and design perspective, the assumption is that by means outside of this system the data to be manipulated within the network element is appropriately partitioned so that any given piece of information is only being manipulated by a single I2RS Client.

Nonetheless, unexpected interactions happen and two (or more) I2RS clients may attempt to manipulate the same piece of data. This is considered an error case. This architecture does not attempt to determine what the right state of data is in such a collision Rather, the architecture mandates that there be decidable means by which I2RS Agents will handle the collisions. The current recommendation is to have a simple priority associated with each I2RS clients, and the highest priority change remains in effect. In the case of priority ties, the first client whose attribution is associated with the data will keep control

In order for this to be useful for I2RS Clients, it is important that it be possible for an I2RS Client to register for changes to any I2RS manipulatable data that it may care about. The I2RS client may then respond to the situation as it sees fit.

6.9. Transactions

In the interest of simplicity, the I2RS architecture does not include multi-message atomicity and rollback mechanisms. Rather, it includes a small range of error handling for a set of operations included in a single message. An I2RS Client may indicate one of the following three error handling for a given message with multiple operations which it sends to an I2RS Agent:

Perform all or none: This traditional SNMP semantic indicates that other I2RS agent will keep enough state when handling a single message to roll back the operations within that message. Either all the operations will succeed, or none of them will be applied and an error message will report the single failure which caused the not to be applied. This is useful when there are, for example, mutual dependencies across operations in the message.

Perform until error: In this case, the operations in the message are applied in the specified order. When an error occurs, no further operations are applied, and an error is returned indicating the failure. This is useful if there are dependencies among the operations and they can be topologically sorted.

Perform all storing errors: In this case, the I2RS Agent will attempt to perform all the operations in the message, and will return error indications for each one that fails. This is useful when there is no dependency across the operation, or where the client would prefer to sort out the effect of errors on its own.

In the interest of robustness and clarity of protocol state, the protocol will include an explicit reply to modification operations even when they fully succeed.

7. Manageability Considerations

Manageability plays a key aspect in I2RS. Some initial examples include:

Resource Limitations: Using I2RS, applications can consume resources, whether those be operations in a time-frame, entries in the RIB, stored operations to be triggered, etc. The ability to set resource limits based upon authorization is important.

Configuration Interactions: The interaction of state installed via the I2RS and via a router's configuration needs to be clearly defined. As described in this architecture, a simple priority that is configured can be used to express the desired policy.

8. Security Considerations

This framework describes interfaces that clearly require serious consideration of security. The ability to identify, authenticate and authorize applications that wish to install state is necessary and briefly described in Section 3.4. Security of communications from the applications is also required as discussed in Section 6.1. Scopes for reading and writing data specified in the context of the data models and the value ranges are discussed briefly in Section 6.4.

9. IANA Considerations

This document includes no request to IANA.

10. Acknowledgements

Significant portions of this draft came from draft-ward-i2rs-framework-00 and draft-atlas-i2rs-policy-framework-00.

The authors would like to thank Nitin Bahadur, Shane Amante, Ed Crabbe, Ken Gray, Carlos Pignataro, Wes George, Joe Clarke, Juergen Schoenwalder, and Jamal Hadi Salim for their suggestions and review.

11. Informative References

[I-D.atlas-i2rs-problem-statement]
Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Problem Statement", draft-atlas-i2rs-problem-statement-01 (work in progress), July 2013.

Authors' Addresses

Alia Atlas
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: akatlas@juniper.net

Joel Halpern
Ericsson

Email: Joel.Halpern@ericsson.com

Susan Hares
ADARA

Email: shares@ndzh.com

Dave Ward
Cisco Systems
Tasman Drive
San Jose, CA 95134
USA

Email: wardd@cisco.com

Thomas D. Nadeau
Juniper Networks

Email: tnadeau@juniper.net