Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 16, 2014

A. Atlas
Juniper Networks
J. Halpern
Ericsson
S. Hares
Hickory Hill Consulting
D. Ward
Cisco Systems
T. Nadeau
Brocade

February 12, 2014

An Architecture for the Interface to the Routing System
draft-ietf-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 August 16, 2014.

Copyright Notice

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, and BGP to exchange protocol state and other information about the state of the network with other routers.

Routers know how to convert all of this information into the forwarding operations that are installed in the forwarding plane. The forwarding plane and the specified forwarding operations then contain active state information that describes the expected and observed operational behavior of the router and which is also needed by the network applications. Network-oriented applications require easy access to this information to learn the network topology, to verify that programmed state is installed in the forwarding plane, to measure the behavior of various flows, routes or forwarding entries, as well as to understand the configured and active states 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 observation of the routing-related state (for example, a Routing Element RIB manager's state), as well as enabling network-oriented 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. This I2RS architecture 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 desired.

Although the I2RS architecture is general enough to support information and data models for a variety of data, the I2RS, and therefore this document, are specifically focused on an interface for routing data.

1.1. Drivers for the I2RS Architecture

There are four key drivers that shape the I2RS architecture. First is the need for an interface that is programmatic, asynchronous, and offers fast, interactive access. Second is the access to structured information and state that is frequently not directly configurable or modeled in existing implementations or configuration protocols. Third is the ability to subscribe to structured, filterable event notifications from the router. Fourth, the operation of I2RS is to 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.ietf-i2rs-problem-statement].

The I2RS facilitates obtaining information from the router. The I2RS provides the ability to not only read specific information, but also to subscribe to targeted information streams and filtered and thresholded events.

Such an interface also facilitates the injection of ephemeral state into the routing system. A non-routing protocol or application could inject state into a routing element via the state-insertion functionality 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). I2RS will only permit modification of state that would be safe, conceptually, to modify via local configuration; no direct manipulation of protocol-internal dynamically determined data is envisioned.

1.2. Architectural Overview

Figure 1 shows the basic architecture for I2RS between applications using I2RS, their associated I2RS Clients, and I2RS Agents. Applications access I2RS services through I2RS clients. A single client can provide access to one or more applications. In the figure, Clients A and B provide access to a single application, while Client P provides access to multiple applications.

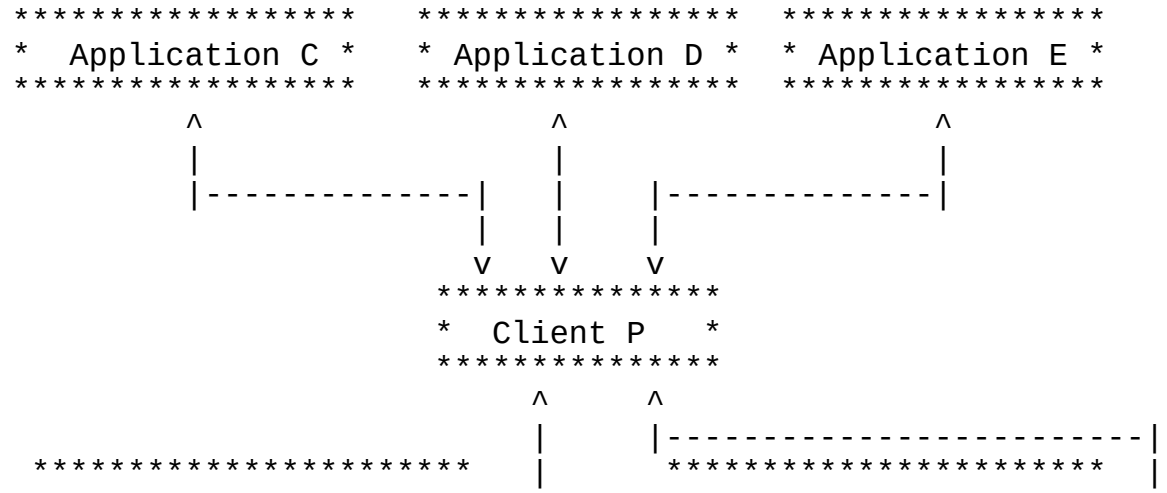
Applications can access I2RS services through local or remote clients. In the figure, Applicatons A and B access I2RS services through local clients, while Applications C, D and E access I2RS services through a remote client.

An I2RS Client can access one or more I2RS agents. In the figure, Clients B and P access I2RS Agents 1 and 2. Likewise, an I2RS Agent can provide service to one or more clients. In the figure, I2RS Agent 1 provides services to Clients A, B and P while Agent 2 provides services to only Clients B and P.

I2RS agents and clients communicate with one another using an asynchronous protocol. Therefore, a single client can post multiple simultaneous requests, either to a single agent or to multiple agents. Furthermore, an agent can process multiple requests, either from a single client or from multiple clients, simultaneously.

The I2RS agent provides read and write access to selected data on the routing element that are organized into I2RS Services.
Section Section 4 describes how access is mediated by authentication and access control mechanisms. In addition to read and write access, the I2RS agent allows clients to subscribe to different types of notifications about events affecting different object instances. An example not related to the creation, modification or deletion of an object instance is when a next-hop in the RIB is resolved enough to be used or when a particular route is selected by the RIB Manager for installation into the forwarding plane. Please see Section 7.6 and Section 7.7 for details.

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.



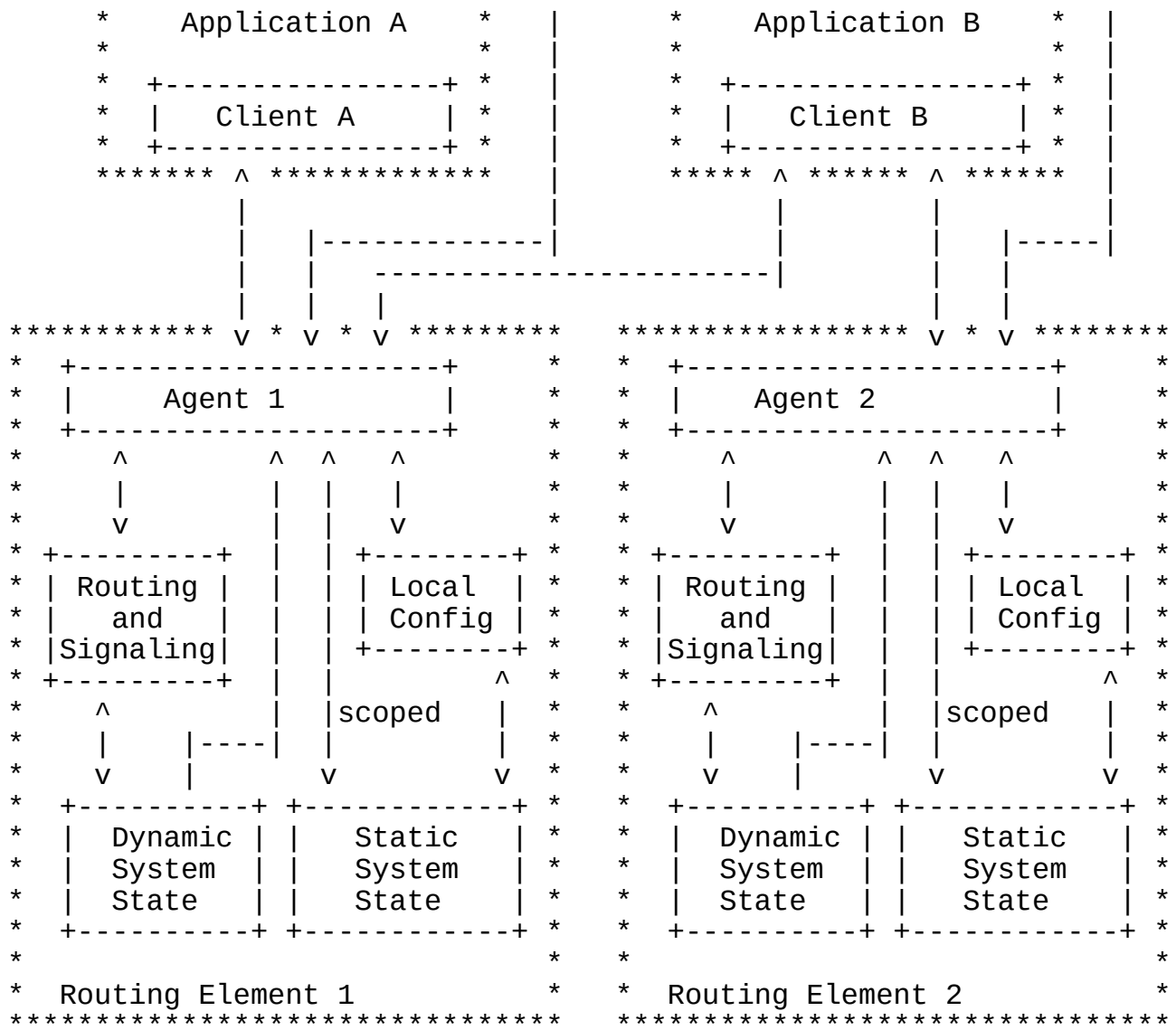


Figure 1: Architecture of I2RS clients and agents

Routing Element: A Routing Element implements some subset 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 and Signaling: 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. This is the subset of operational state that is needed by network applications based on I2RS that is not contained in the routing and signaling information. 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.

Static System State: An I2RS agent needs access to static state on a routing element beyond what is contained in the routing subsystem. An example of such state is specifying queueing behavior for an interface or traffic. How the I2RS agent modifies or obtains this information is out of scope, but the standardized information and data models for what is exposed are part of I2RS.

I2RS Agent: See the definition in Section 2.

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

I2RS Client: See the definition in Section 2.

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. 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.

In contrast, although 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 7.8. 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 6.3.

In addition it must be noted that there may be indirect interactions between write operations. A tivial example of this is when two different but overlapping prefixes are written with different forwarding behavior. Detection and avoidance of such interactions is outside the scope of the I2RS work and is left to agent design and implementation.

2. Terminology

The following terminology is used in this document.

agent or I2RS Agent: An I2RS agent provides the supported I2RS services from the local system’s routing sub-systems by interacting with the routing element to provide specified behavior. The I2RS agent understands the I2RS protocol and can be contacted by I2RS clients.

client or I2RS Client: A client implements the I2RS protocol, uses it to communicate with I2RS Agents, and uses the I2RS services to accomplish a task. 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 I2RS agents to modify the state of the routing system the client interacts with to achieve operational goals. 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. The read scope specifies the access restrictions to both see the existence of data and read the value of that data.

notification scope: The set of events and associated information that the I2RS Client can request be pushed by the I2RS Agent. I2RS Clients have the ability to register for specific events and information streams, but must be constrained by the access restrictions associated with their notification scope.

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, write scope, or notification scope, the term scope applies to the read scope, write scope, and notification 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 may 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 available 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 quantity and diversity 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 deployable and robust. Complexity in the protocol hinders implementation, robustness, and deployability. Also, data models complexity may complicate extensibility.

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

Naturally, extensibility of the protocol and data model is very important. In particular, given the necessary scope limitations of the initial work, it is critical that the initial design include strong support for extensibility.

The scope of the I2RS work is being restricted in the interests of achieving a deliverable and deployable result. The I2RS Working Group is modeling only a subset of the data of interest. It is clearly desirable for the data models defined in the I2RS to be useful 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. This reinforces the criticality of

designing the data models to be highly extensible, preferably in a regular and simple fashion.

The I2RS Working Group is defining operations for the I2RS protocol. It would be optimistic to assume that more and different ones may not be needed when the scope of I2RS increases. Thus, it is important to consider extensibility not only of the underlying services' data models, but also of the primitives and protocol operations.

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.

4. Security Considerations

This I2RS architecture describes interfaces that clearly require serious consideration of security. First, here is a brief description of the assumed security environment for I2RS. The I2RS Agent associated with a Routing Element is a trusted part of that Routing Element. For example, it may be part of a vendor-distributed signed software image for the entire Routing Element or it may be trusted signed application that an operator has installed. The I2RS Agent is assumed to have a separate authentication and authorization channel by which it can validate both the identity and permissions associated with an I2RS Client. To support numerous and speedy interactions between the I2RS Agent and I2RS Client, it is assumed that the I2RS Agent can also cache that particular I2RS Clients are trusted and their associated authorized scope. This implies that either in a pull model, the permission information may be old until the I2RS Agent rerequests it, or in a push model, that the

authentication and authorization channel can notify the I2RS Agent of changes.

An I2RS Client is not automatically trustworthy. It has identity information and applications using that I2RS Client should be aware of the scope limitations of that I2RS Client. If the I2RS Client is acting as a broker for multiple applications, managing the security, authentication and authorization for that communication is out of scope; nothing prevents I2RS and a separate authentication and authorization channel from being used. Regardless of mechanism, an I2RS Client that is acting as a broker is responsible for determining that applications using it are trusted and permitted to make the particular requests.

Different levels of integrity, confidentiality, and replay protection are relevant for different aspects of I2RS. The primary communication channel that is used for client authentication and then used by the client to write data requires integrity, privacy and replay protection. Appropriate selection of a default required transport protocol is the preferred way of meeting these requirements.

Other communications via I2RS will not require integrity, confidentiality, and replay protection. For instance, if an I2RS Client subscribes to an information stream of prefix announcements from OSPF, those may require integrity but probably not confidentiality or replay protection. Similarly, an information stream of interface statistics may not even require guaranteed delivery. In Section 7.2, more reasoning for multiple communication channels is provided. From the security perspective, it is critical to realize that an I2RS Agent may open a new communication channel based upon information provided by an I2RS Client; to avoid an indirect attack, such a request must be done in the context of an authenticated and authorized client whose communications cannot have been altered.

4.1. Identity and Authentication

As discussed above, all control exchanges between the I2RS client and agent should be authenticated and integrity protected (such that the contents cannot be changed without detection). Further, 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 clients may be operating on behalf of other applications. While those applications' identities are not needed for authentication or

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.

4.2. Authorization

All operations using I2RS, both observation and manipulation, should be subject to appropriate authorization controls. Such authorization is based on the identity and assigned role of the I2RS client performing the operations and the I2RS agent in the network element.

I2RS Agents, in performing information collection and manipulation, will be acting on behalf of the I2RS clients. As such, each operation authorization will be based on the lower of the two permissions of the agent itself and of the authenticated client. The mechanism by which this authorization is applied within the device is outside of the scope of I2RS.

The appropriate or necessary level of granularity for scope can depend upon the particular I2RS Service and the implementation's granularity. An approach to a similar access control problem is defined in the NetConf Access Control Model[RFC6536]; it allows arbitrary access to be specified for a data node instance identifier while defining meaningful manipulable defaults. The ability to specify one or more groups or roles that a particular I2RS Client belongs and then define access controls in terms of those groups or roles is expected. When a client is authenticated, its group or role membership should be provided to the I2RS Agent. The set of access control rules that an I2RS Agent uses would need to be either provided via Local Config, exposed as an I2RS Service for manipulation by authorized clients, or via some other method.

5. Network Applications and I2RS Client

I2RS is expected to be used by network-oriented applications in different architectures. While the interface between a network-oriented application and the I2RS client is outside the scope of I2RS, considering the different architectures is important to sufficiently specify I2RS.

In the simplest architecture, a network-oriented application has an I2RS client as a library or driver for communication with routing elements.

In the broker architecture, multiple network-oriented applications communicate in an unspecified fashion to a broker application that contains an I2RS Client. That broker application requires additional

functionality for authentication and authorization of the network-oriented applications; such functionality is out of scope for I2RS but similar considerations to those described in Section 4.2 do apply. As discussed in Section 4.1, the broker I2RS Client should determine distinct opaque identifiers for each network-oriented application that is using it. The the broker I2RS Client can pass along the appropriate value as a secondary identifier which can be used for tracking attribution of operations.

In the third architecture, a routing element or network-oriented application that uses an I2RS Client to access services on a different routing element may also contain an I2RS agent to provide services to other network-oriented applications. However, where the needed information and data models for those services differs from that of a conventional routing element, those models are, at least initially, out of scope for I2RS. Below is an example of such a network application

5.1. Example Network Application: 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, such as interface and label-switched path (LSP) information. In addition, the Topology Manager may collect link-state data in several ways - either via I2RS models, by peering with BGP-LS[I-D.ietf-idr-ls-distribution] or listening into the IGP.

The set of functionality and collected information that is the Topology Manager may be embedded as a component of a larger application, such as a path computation application. As a stand-alone application, the Topology Manager could be useful to other network applications by providing a coherent picture of the network state accessible via another interface. That interface might use the same I2RS protocol and could provide a topology service using extensions to the I2RS data models.

6. 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.

6.1. 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.

6.2. I2RS State Storage

State modification requests are sent to the I2RS agent in a routing element by I2RS clients. The I2RS agent is responsible for applying these changes to the system, subject to the authorization discussed above. The I2RS agent will retain knowledge of the changes it has applied, and the client on whose behalf it applied the changes. The I2RS agent will also store active subscriptions. These sets of data form the I2RS data store. This data is retained by the agent until the state is removed by the client, overridden by some other operation such as CLI, or the device reboots. Meaningful logging of the application and removal of changes is recommended. I2RS applied changes to the routing element state will not be retained across routing element reboot. The I2RS data store is not preserved across routing element reboots; thus the I2RS agent will not attempt to reapply such changes after a reboot.

6.2.1. I2RS Agent Failure

If it is possible for an I2RS Agent to fail independently of the associated routing element, the behavior for any associated ephemeral I2RS state needs to be clearly described. The I2RS state should be preserved until the associated routing element has itself rebooted or until the I2RS state is explicitly torn down. This is desirable since the I2RS Client has no way of learning that an I2RS Agent has unexpected failed until that I2RS Agent has restarted; in the interval between failure and recovery, the I2RS Client will be assuming that its ephemeral state remains. If failure of the I2RS agent causes the ephemeral I2RS state to be removed, then this should be indicated via a capability.

There are two different failure types that are possible and each has different behavior.

Unexpected failure: In this case, the I2RS Agent has unexpectedly crashed and thus cannot notify its clients of anything. If an I2RS Agent can crash separately from its associated routing element, then that I2RS Agent must cache each known I2RS Client. When an I2RS Agent starts, it notifies each saved I2RS Client that the I2RS Agent is up and includes an agent-boot-count that indicates how many times the I2RS Agent has restarted since the associated routing element restarted. The agent-boot-count allows an I2RS Client to determine if the I2RS Agent has restarted; if so, the I2RS Client may need to resubscribe to notifications and information streams. The I2RS Agent should also indicate whether the I2RS ephemeral state was preserved in the Routing Element.

Graceful failure: In this case, the I2RS Agent can do specific limited work as part of the process of being disabled. First, the I2RS Agent can optionally notify all its clients that their state is being torn down; if no such notification is sent, then that ephemeral state is not torn down. Second, the I2RS Agent must notify all its cached clients that the agent is going down.

6.2.2. Starting and Ending

When an I2RS client applies changes via the I2RS protocol, those changes are applied and left until removed or the routing element reboots. The network application may make decisions about what to request via I2RS based upon a variety of conditions that imply different start times and stop times. That complexity is managed by the network application and is not handled by I2RS.

6.2.3. 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, subject to its notification scope, regarding state modification or removal by a particular I2RS Client.

6.3. Interactions with Local Config

Changes may originate from either Local Config or from I2RS. The modifications and data stored by I2RS are separate from the local device configuration, but conflicts between the two must be resolved in a deterministic manner that respects operator-applied policy. That policy can determine whether Local Config overrides a particular I2RS client's request or vice versa. To achieve this end, either by default Local Config always wins or, optionally, a routing element may permit a priority to be configured on the device for the Local Config mechanism. The policy mechanism in the later case is comparing the I2RS client's priority with that priority assigned to the Local Config.

When the Local Config always wins, some communication between that subsystem and the I2RS Agent is still necessary. That communication contains the details of each specific device configuration change that the I2RS Agent is permitted to modify. In addition, when the system determines, that a client's I2RS state is preempted, the I2RS agent must notify the affected I2RS agents; how the system determines this is implementation-dependent.

It is critical that policy based upon the source is used because the resolution cannot be time-based. Simply allowing the most recent state to prevail could cause race conditions where the final state is not repeatably deterministic.

6.4. Routing Components and Associated I2RS Services

For simplicity, each logical protocol or set of functionality that can 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. When a full implementation is not mandatory, an I2RS Service should include a capability model so that implementations can indicate which parts of the service are supported. Each I2RS Service requires an information model that describes at least the following: data that can be read, data that can be written, notifications that can be subscribed to, and the capability model mentioned above.

The initial services included in the I2RS architecture are as follows.

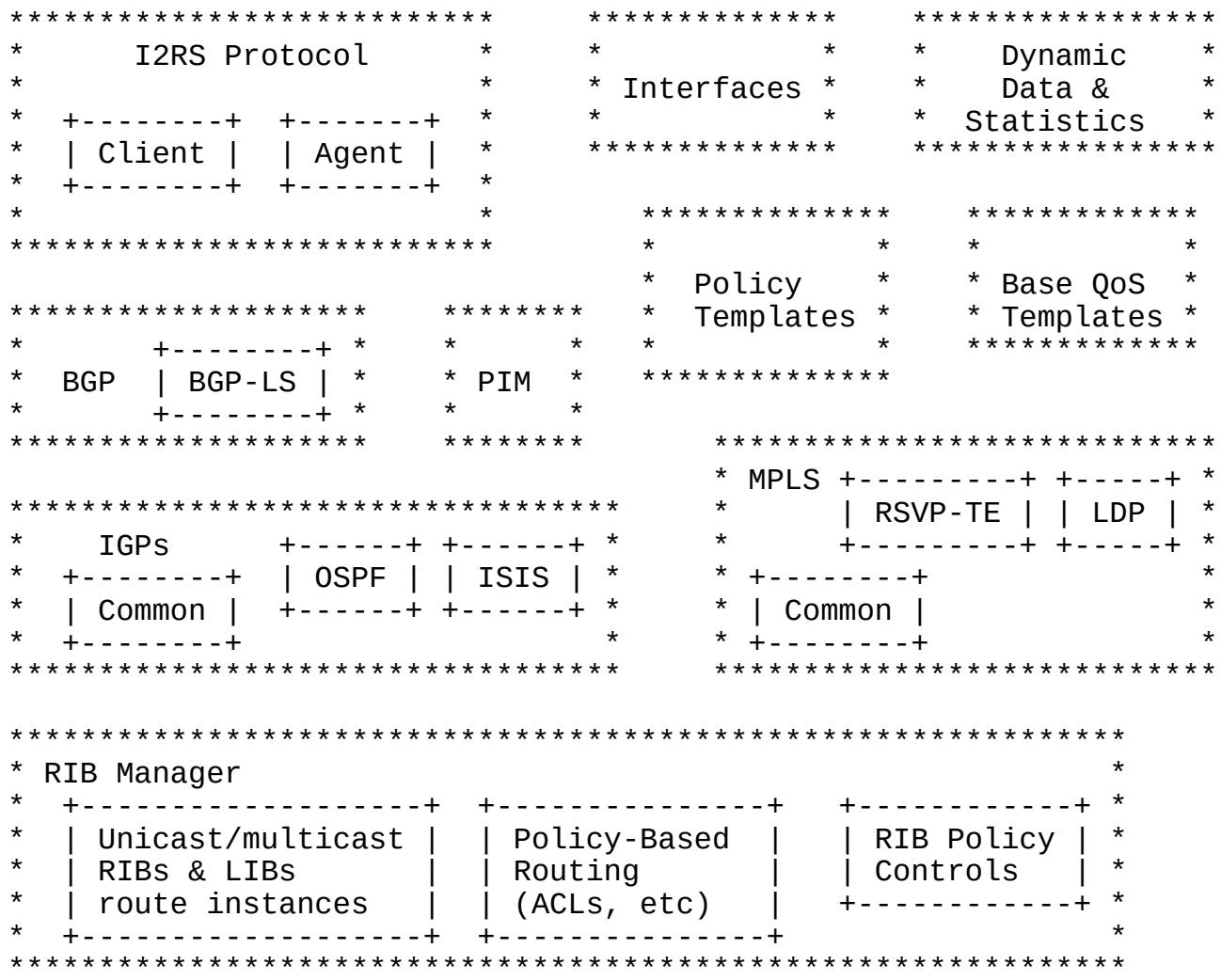


Figure 2: Anticipated I2RS Services

There are relationships between different I2RS Services - whether those be the need for the RIB to refer to specific interfaces, the desire to refer to common complex types (e.g. links, nodes, IP addresses), or the ability to refer to implementation-specific functionality (e.g. pre-defined templates to be applied to interfaces or for QoS behaviors that traffic is direct into).
Section Section 6.4.5 discussing information modeling constructs and the range of relationship types that are applicable.

6.4.1. Routing and Label Information Bases

Routing elements may maintain one or more Information Bases. Examples include Routing Information Bases such as IPv4/IPv6 Unicast or IPv4/IPv6 Multicast. Another such example includes the MPLS Label Information Bases, per-platform- or per-interface." This

functionality, exposed via an I2RS Service, must interact smoothly with the same mechanisms that the routing element already uses to handle RIB input from multiple sources, so as to safely change the system state. Conceptually, this can be handled by having the I2RS Agent communicate with a RIB Manager as a separate routing source.

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

6.4.2. IGPs, BGP and Multicast Protocols

A separate I2RS Service can expose each routing protocol on the device. Such I2RS services may include 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 to 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

- subscribing to an information stream of route changes

- receiving notifications about peers coming up or going down

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 MUST NOT allowed in order to preserve network state consistency.

6.4.3. MPLS

I2RS Services will be needed to expose 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). This should include all local information about LSPs originating in, transiting, or terminating in this Routing Element.

6.4.4. 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 information models and supported in the expected interfaces between the I2RS Agent and the network element, in order to provide basic capabilities and the hooks for future extensibility.

By taking advantage of extensibility and sub-classing, information models can specify use of a basic model that can be replaced by a more detailed model.

6.4.5. Information Modeling, Device Variation, and Information Relationships

I2RS depends heavily on information models of the relevant aspects of the Routing Elements to be manipulated. These models drive the data models and protocol operations for I2RS. It is important that these informational models deal well with a wide variety of actual implementations of Routing Elements, as seen between different products and different vendors. There are three ways that I2RS information models can address these variations: class or type inheritance, optional features, and templating.

6.4.5.1. Managing Variation: Object Classes/Types and Inheritance

Information modeled by I2RS from a Routing Element can be described in terms of classes or types or object. Different valid inheritance definitions can apply. What is appropriate for I2RS to use is not determined in this architecture; for simplicity, class and subclass will be used as the example terminology. This I2RS architecture does require the ability to address variation in Routing Elements by allowing information models to define parent or base classes and subclasses.

The base or parent class defines the common aspects that all Routing Elements are expected to support. Individual subclasses can represent variations and additional capabilities. When applicable, there may be several levels of refinement. The I2RS protocol can then provide mechanisms to allow an I2RS client to determine which classes a given I2RS Agent has available. Clients which only want basic capabilities can operate purely in terms of base or parent

classes, while a client needing more details or features can work with the supported sub-class(es).

As part of I2RS information modeling, clear rules should be specified for how the parent class and subclass can relate; for example, what changes a subclass can make to its parent? The description of such rules should be done so that it can apply across data modeling tools until the I2RS data modeling language is selected.

6.4.5.1.1. Managing Variation: Optionality

I2RS Information Models must be clear about what aspects are optional. For instance, must an instance of a class always contain a particular data field X? If so, must the client provide a value for X when creating the object or is there a well-defined default value? From the Routing Element perspective, in the above example, is support of X required so that values for X can be accepted and processed? If not, how does the I2RS client determine whether the I2RS agent can accept and apply values for X?

Optional behavior can also be extended to the ranges of values a given piece of information can take, the length of strings, the existence of particular events, and other aspects of information. The information model needs to be clear about what is required of the clients, what is required of agents, and what is permitted to each one.

6.4.5.1.2. Managing Variation: Templating

A template is a collection of information to address a problem; it cuts across the notions of class and object instances. A template provides a set of defined values for a set of information fields and can specify a set of values that must be provided to complete the template. Further, a flexible template scheme may that some of the defined values can be over-written.

For instance, assigning traffic to a particular service class might be done by specifying a template Queueing with a parameter to indicate Gold, Silver, or Best Effort. The details of how that is carried out are not modeled. This does assume that the necessary templates are made available on the Routing Element via some mechanism other than I2RS. The idea is that by providing suitable templates for tasks that need to be accomplished, with templates implemented differently for different kinds of Routing Elements, the client can easily interact with the Routing Element without concern for the variations which are handled by values included in the template.

If implementation variation can be exposed in other ways, templates may not be needed. However, templates themselves could be objects referenced in the protocol messages, with Routing Elements being configured with the proper templates to complete the operation. This is a topic for further discussion.

6.4.5.1.3. Object Relationships

Objects (in a Routing Element or otherwise) do not exist in isolation. They are related to each other. One of the important things a class definition does is represent the relationships between instances of different classes. These relationships can be very simple, or quite complicated. The following lists the information relationships that the information models need to support. [[Editors' note: All of these are for discussion, and it is expected that the list may be changed during WG discussion.]]

6.4.5.1.3.1. Initialization

The simplest relationship is that one object instances is initialized by copying another. For example, one may have an object instance that represents the default setup for a tunnel, and all new tunnels have fields copied from there if they are not set as part of establishment. This is closely related to the templates discussed above, but not identical. Since the relationship is only momentary it is often not formally represented in modeling, but only captured in the semantic description of the default object.

6.4.5.1.3.2. Correlation Identification

Often, it suffices to indicate in one object that it is related to a second object, without having a strong binding between the two. So an Identifier is used to represent the relationship. This can be used to allow for late binding, or a weak binding that does not even need to exist. A policy name in an object might indicate that if a policy by that name exists, it is to be applied under some circumstance. In modeling this is often represented by the type of the value.

6.4.5.1.3.3. Object References

Sometimes the relationship between objects is stronger. A valid ARP entry has to point to the active interface over which it was derived. This is the classic meaning of an object reference in programming. It can be used for relationships like containment or dependence. This is usually represented by an explicit modeling link.

6.4.5.1.3.4. Active Reference

There is an even stronger form of coupling between objects if changes in one of the two objects are always to be reflected in the state of the other. For example, if a Tunnel has an MTU, and link MTU changes need to immediately propagate to the Tunnel MTU, then the tunnel is actively coupled to the link interface. This kind of active state coupling implies some sort of internal bookkeeping to ensure consistency, often conceptualized as a subscription model across objects.

7. I2RS Client Agent Interface

7.1. One Control and Data Exchange Protocol

This I2RS Architecture presumes that there is one I2RS protocol for control and data exchange. This helps meet the goal of simplicity and thereby enhances deployability. Whether such a protocol is built upon extending existing mechanisms or requires a new mechanism is under active 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.

7.2. Communication Channels

Multiple communication channels and multiple types of communication channels are required. There may be a range of requirements (e.g. confidentiality, reliability), and to support the scaling there may need to be channels originating from multiple sub-components of a routing element and/or to multiple parts of an I2RS client. All such communication channels will use the same higher level protocol. Use of additional channels for communication will be coordinated between the I2RS client and the I2RS agent.

7.3. Capability Negotiation

The support for different protocol capabilities and I2RS Services will vary across I2RS Clients and Routing Elements supporting I2RS Agents. Since each I2RS Service is required to include a capability model (see Section 6.4), negotiation at the protocol level can be restricted to protocol specifics and which I2RS Services are supported.

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.

The protocol capability negotiation can be segmented into the basic version negotiation (required to ensure basic communication), and the more complex capability exchange which can take place within the base protocol mechanisms. In particular, the more complex protocol and mechanism negotiation can be addressed by defining information models for both the I2RS Agent and the I2RS Client. These information models can describe the various capability options. This can then represent and be used to communicate important information about the agent, and the capabilities thereof.

7.4. 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 based on the primary unique identity, the I2RS Client 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.

7.4.1. 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.

7.5. 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.

For many applications, it may be desirable to clean up state if a network application dies before removing the state it has created. Typically, this is dealt with in terms of network application redundancy. If stronger mechanisms are desired, mechanisms outside of I2RS may allow a supervisory network application to monitor I2RS clients, and based on policy known to the supervisor clean up state if applications die. More complex mechanism instantiated in the I2RS agent would add complications to the I2RS protocol and are thus left for future work.

Some examples of such a mechanism include the following. In one option, the client could request state clean-up if a particular transport session is terminated. 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.

7.6. Notifications

As with any policy system interacting with the network, the I2RS Client 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. A notification may also be due to a regular event.

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 I2RS Agents to send notifications to a number of Clients. The I2RS Client should be able to filter the specific notifications that will be received; the specific types of events and filtering operations can vary by information model and need to be specified as part of the information model.

The I2RS information model needs to include representation of these events. As discussed earlier, the capability information in the model will allow I2RS clients to understand which events a given I2RS Agent is capable of generating.

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.

7.7. 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.

7.8. 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 should be when such a collision happens. Rather, the architecture mandates that there be decidable means by which I2RS Agents handle the collisions. The mechanism for this 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 approach to multi-headed control to be useful for I2RS Clients, it is important that it be possible for an I2RS Client to register for changes to any changes made by I2RS to data that it may care about. This is included in the I2RS event mechanisms. This also needs to apply to changes made by CLI/NETCONF/SNMP within the

write-scope of the I2RS Agent, as the same priority mechanism (even if it is "CLI always wins") applies there. The I2RS client may then respond to the situation as it sees fit.

7.9. 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 them 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 or write operations even when they fully succeed.

8. 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 is used to provide sufficient policy flexibility.

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, Ron Bonica, Joe Clarke, Juergen Schoenwalder, Jamal Hadi Salim, Scott Brim, and Thomas Narten for their suggestions and review.

11. Informative References

[I-D.ietf-i2rs-problem-statement] Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Problem Statement", draft-ietf-i2rs-problem-statement-00 (work in progress), August 2013.

[I-D.ietf-idr-ls-distribution] Gredler, H., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and TE Information using BGP", draft-ietf-idr-ls-distribution-04 (work in progress), November 2013.

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

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
Hickory Hill Consulting

Email: shares@ndzh.com

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

Email: wardd@cisco.com

Thomas D. Nadeau
Brocade

Email: tnadeau@lucidvision.com