Internet Engineering Task Force (IETF)
Request for Comments: 7921
Category: Informational
ISSN: 2070-1721

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

June 2016

An Architecture for the Interface to the Routing System

Abstract

This document describes the IETF architecture for a standard, programmatic interface for state transfer in and out of the Internet routing system. It describes the high-level architecture, the building blocks of this high-level architecture, and their interfaces, with particular focus on those to be standardized as part of the Interface to the Routing System (I2RS).

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7921.

Table of Contents

1. Introduction

Routers that form the Internet 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, IS-IS, and BGP to exchange reachability information, topology information, protocol state, and other information about the state of the network with other routers.

Routers convert all of this information into forwarding entries, which are then used to forward packets and flows between network elements. The forwarding plane and the specified forwarding entries then contain active state information that describes the expected and observed operational behavior of the router and that 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. Network-oriented applications also require easy access to an interface, which will allow them to program and control state related to forwarding.

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 routing system. This I2RS architecture recognizes that the routing system and a router's Operating System (OS) provide useful mechanisms that applications could harness to accomplish application-level goals. These network-oriented applications can leverage the I2RS programmatic interface to create new ways to combine retrieving Internet routing data, analyzing this data, and setting state within routers.

Fundamental to I2RS are clear data models that define the semantics of the information that can be written and read. I2RS provides a way for applications to customize network behavior while leveraging the existing routing system as desired. I2RS provides a framework for applications (including controller applications) to register and to request the appropriate information for each particular application.

Although the I2RS architecture is general enough to support information and data models for a variety of data, and aspects of the I2RS solution may be useful in domains other than routing, I2RS and this document are specifically focused on an interface for routing data.

Security is a concern for any new I2RS. Section 4 provides an overview of the security considerations for the I2RS architecture. The detailed requirements for I2RS protocol security are contained in [I2RS-PROT-SEC], and the detailed security requirements for environment in which the I2RS protocol exists are contained in [I2RS-ENV-SEC].

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 and asynchronous and that offers fast, interactive access for atomic operations. 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 [RFC7920].

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

Such an interface also facilitates the injection of ephemeral state into the routing system. Ephemeral state on a router is the state that does not survive the reboot of a routing device or the reboot of the software handling the I2RS software on a routing device. A non-routing protocol or application could inject state into a routing element via the state-insertion functionality of 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 possible 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 I2RS client can provide access to one or more applications. This figure also shows the types of data models associated with the routing system (dynamic configuration, static configuration, Local Configuration, and routing and signaling configuration) that the I2RS agent data models may access or augment.

Figure 1 is similar to Figure 1 in [RFC7920], but the figure in this document shows additional detail on how the applications utilize I2RS clients to interact with I2RS agents. It also shows a logical view of the data models associated with the routing system rather than a functional view (RIB, Forwarding Information Base (FIB), topology, policy, routing/signaling protocols, etc.)

In Figure 1, Clients A and B each provide access to a single application (Applications A and B, respectively), while Client P provides access to multiple applications.

Applications can access I2RS services through local or remote clients. A local client operates on the same physical box as the routing system. In contrast, a remote client operates across the network. In the figure, Applications A and B access I2RS services through local clients, while Applications C, D, and E access I2RS services through a remote client. The details of how applications communicate with a remote client is out of scope for I2RS.

An I2RS client can access one or more I2RS agents. In Figure 1, Clients B and P access I2RS agents 1 and 2. Likewise, an I2RS agent can provide service to one or more clients. In this 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 4 describes how access is mediated by authentication and access control mechanisms. Figure 1 shows I2RS agents being able to write ephemeral static state (e.g., RIB entries) and to read from dynamic static (e.g., MPLS Label Switched Path Identifier (LSP-ID) or number of active BGP peers).

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. One example of a notification of such an event (which is unrelated to an object creation, modification or deletion) is when a next hop in the RIB is resolved in a way that allows it to be used by a RIB manager for installation in the forwarding plane as part of a particular route. Please see Sections 7.6 and 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.

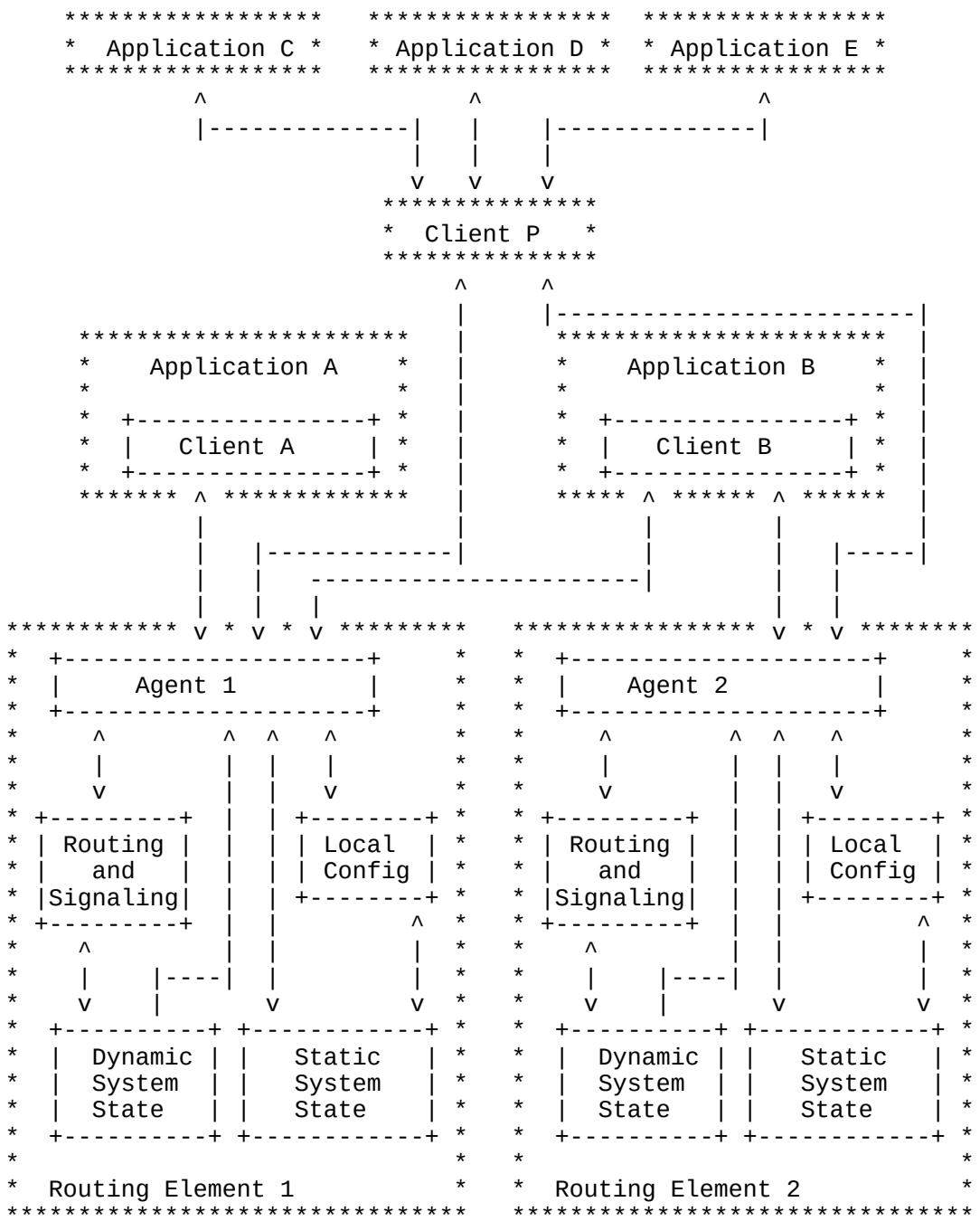

Figure 1: Architecture of I2RS Clients and Agents

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 IS-IS, OSPF, BGP, PIM, etc.,

* A BGP speaker acting as a Route Reflector,

* A Label Switching Router (LSR) that implements RSVP-TE, OSPF-TE, and the Path Computation Element (PCE) Communication Protocol (PCEP) and has a forwarding plane and associated RIB Manager, and

* A server that runs IS-IS, OSPF, and BGP and uses Forwarding and Control Element Separation (ForCES) to control a remote forwarding plane.

A Routing Element may be locally managed, whether via command-line interface (CLI), SNMP, or the Network Configuration Protocol (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 Configuration: The black box behavior for interactions between the ephemeral state that I2RS installs into the routing element; Local Configuration is defined by this document and the behaviors specified by the I2RS protocol.

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, flow data, 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. 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, two clients should not attempt to write (modify) the same piece of information on an I2RS agent. This is considered an error. However, such collisions may happen and Section 7.8 ("Multi-headed Control") describes how the I2RS agent resolves collision by first utilizing priority to resolve collisions and second by servicing the requests in a first-in, first-served basis. The I2RS architecture includes this definition of behavior for this case simply for predictability, not because this is an intended result. This predictability will simplify error handling and suppress oscillations. If additional error cases beyond this simple treatment are required, these error cases 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 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 basic 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 subsystems 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 their associated routing systems 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 read scope of an I2RS client within an I2RS agent is the set of information that the I2RS client is authorized to read within the I2RS agent. The read scope specifies the access restrictions to both see the existence of data and read the value of that data.

notification scope: The notification scope is 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 write scope is the set of field values that 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. If an identity has multiple roles in the security system, the identity is permitted to perform any operations any of those roles permit. Multiple identities may use the same security role.

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.

identity and scope: A single identity can be associated with multiple roles. Each role has its own scope, and an identity associated with multiple roles can use the combined scope of all its roles. More formally, each identity has:

* a read scope that is the logical OR of the read scopes associated with its roles,

* a write scope that is the logical OR of the write scopes associated with its roles, and

* a notification scope that is the logical OR of the notification scopes associated with its roles.

secondary identity: An I2RS client may supply a secondary opaque identifier for a secondary identity that is not interpreted by the I2RS agent. An example of the use of the secondary opaque identifier 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.

ephemeral data: Ephemeral data is data that does not persist across a reboot (software or hardware) or a power on/off condition. Ephemeral data can be configured data or data recorded from operations of the router. Ephemeral configuration data also has the property that a system cannot roll back to a previous ephemeral configuration state.

group: The NETCONF Access Control Model [RFC6536] uses the term "group" in terms of an administrative group that supports the well-established distinction between a root account and other types of less-privileged conceptual user accounts. "Group" still refers to a single identity (e.g., root) that is shared by a group of users.

routing system/subsystem: A routing system or subsystem is a set of software and/or hardware that determines where packets are forwarded. The I2RS agent is a component of a routing system. The term "packets" may be qualified to be layer 1 frames, layer 2 frames, or layer 3 packets. The phrase "Internet routing system" implies the packets that have IP as layer 3. A routing "subsystem" indicates that the routing software/hardware is only the subsystem of another larger system.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Key Architectural Properties

Several key architectural properties for the I2RS protocol are elucidated below (simplicity, extensibility, and model-driven programmatic interfaces). However, some architectural properties such as performance and scaling are not described below because they are discussed in [RFC7920] and because they may vary based on the particular use cases.

3.1. Simplicity

There have been many efforts over the years to improve 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.

While the types of operations contemplated here are complex in their nature, it is critical that I2RS be easily deployable and robust. Adding complexity beyond what is needed to satisfy well known and understood requirements would hinder the ease of implementation, the robustness of the protocol, and the deployability of the protocol. Overly complex data models tend to ossify information sets by attempting to describe and close off every possible option, complicating extensibility.

Thus, one of the key aims for I2RS is to 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?" If we need the complexity, we should ask ourselves, "Is this the simplest way to provide this complexity in the I2RS external interface?"

3.2. Extensibility

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 I2RS work is being designed in phases to provide deliverable and deployable results at every phase. Each phase will have a specific set of requirements, and the I2RS protocol and data models will progress toward these requirements. Therefore, it is clearly desirable for the I2RS data models to be easily and highly extensible to represent additional aspects of the network elements or network systems. It should be easy to integrate data models from I2RS with other data. 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 data-model-focused processing of management data that provides modular implementation in I2RS clients and I2RS agents. The I2RS client only needs to implement the models the I2RS client is able to access. The I2RS agent only needs to implement the data models the I2RS agent supports.

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.

The purpose of the data model is to provide a definition of the information regarding the routing system that can be used in operational networks. If routing information is being modeled for the first time, a logical information model may be standardized prior to creating the data model.

4. Security Considerations

This I2RS architecture describes interfaces that clearly require serious consideration of security. As an architecture, I2RS has been designed to reuse existing protocols that carry network management information. Two of the existing protocols that are being reused for the I2RS protocol version 1 are NETCONF [RFC6241] and RESTCONF [RESTCONF]. Additional protocols may be reused in future versions of the I2RS protocol.

The I2RS protocol design process will be to specify additional requirements (including security) for the existing protocols in order in order to support the I2RS architecture. After an existing protocol (e.g., NETCONF or RESTCONF) has been altered to fit the I2RS requirements, then it will be reviewed to determine if it meets these requirements. During this review of changes to existing protocols to serve the I2RS architecture, an in-depth security review of the revised protocol should be done.

Due to the reuse strategy of the I2RS architecture, this security section describes the assumed security environment for I2RS with additional details on a) identity and authentication, b) authorization, and c) client redundancy. Each protocol proposed for

inclusion as an I2RS protocol will need to be evaluated for the security constraints of the protocol. The detailed requirements for the I2RS protocol and the I2RS security environment will be defined within these global security environments.

The I2RS protocol security requirements for I2RS protocol version 1 are contained in [I2RS-PROT-SEC], and the global I2RS security environment requirements are contained [I2RS-ENV-SEC].

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 a 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 the permission information may be old either in a pull model until the I2RS agent re-requests it or in a push model until the authentication and authorization channel can notify the I2RS agent of changes.

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

An I2RS client is not automatically trustworthy. Each I2RS client is associated with an identity with a set of scope limitations. Applications using an I2RS client should be aware that the scope limitations of an I2RS client are based on its identity (see Section 4.1) and the assigned role that the identity has. A role sets specific authorization limits on the actions that an I2RS client can successfully request of an I2RS agent (see Section 4.2). For example, one I2RS client may only be able to read a static route table, but another client may be able add an ephemeral route to the static route table.

If the I2RS client is acting as a broker for multiple applications, then managing the security, authentication, and authorization for that communication is out of scope; nothing prevents the broker from using the I2RS protocol and a separate authentication and authorization channel from being used. Regardless of the 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, confidentiality and replay protection. Appropriate selection of a default required transport protocol is the preferred way of meeting these requirements.

Other communications via I2RS may 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, additional logins regarding multiple communication channels and their use 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 (as described in Section 7.2). For example, an I2RS client may request notifications of certain events, and the agent will open a communication channel to report such events. Therefore, 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 trade-offs 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 an application identifier (and from that to the application's identity). This tracking of operations to an application supports I2RS functionality for tracing actions (to aid troubleshooting in routers) and logging of network changes.

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. Multiple identities may use the same role(s). As noted in the definitions of "identity" and "role" above, if multiple roles are associated with an identity then the identity is authorized to perform any operation authorized by any of its roles.

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 (NACM) [RFC6536]; it allows arbitrary access to be specified for a data node instance identifier while defining meaningful manipulable defaults. The identity within NACM [RFC6536] can be specified as either a user name or a group user name (e.g., Root), and this name is linked a scope policy that is contained in a set of access control rules. Similarly, it is expected the I2RS identity links to one role that has a scope policy specified by a set of access control rules. This scope policy can be provided via Local Configuration, exposed as an I2RS service for manipulation by authorized clients, or via some other method (e.g., Authentication, Authorization, and Accounting (AAA) service)

While the I2RS agent allows access based on the I2RS client's scope policy, this does not mean the access is required to arrive on a particular transport connection or from a particular I2RS client by the I2RS architecture. The operator-applied scope policy may or may not restrict the transport connection or the identities that can access a local I2RS agent.

When an I2RS client is authenticated, its identity is provided to the I2RS agent, and this identity links to a role that links to the scope policy. Multiple identities may belong to the same role; for example, such a role might be an Internal-Routes-Monitor that allows reading of the portion of the I2RS RIB associated with IP prefixes used for internal device addresses in the AS.

4.3. 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 the need for troubleshooting and logging of network changes to be informed about which network application is actually active. At a minimum, basic transport information about each connection and time can be logged with the identity.

4.4. I2RS in Personal Devices

If an I2RS agent or I2RS client is tightly correlated with a person (such as if an I2RS agent is running on someone's phone to control tethering), then this usage can raise privacy issues, over and above the security issues that normally need to be handled in I2RS. One example of an I2RS interaction that could raise privacy issues is if the I2RS interaction enabled easier location tracking of a person's phone. The I2RS protocol and data models should consider if privacy issues can arise when clients or agents are used for such use cases.

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 of direct access, 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 broker I2RS client can pass along the appropriate value as a secondary identifier, which can be used for tracking attribution of operations.

In a 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. The following section describes 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 -- via I2RS models, by peering with BGP-LS [RFC7752], or by 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. There may be multiple I2RS agents within the same router. In such a case, they must have non-overlapping sets of information that they manipulate.

To facilitate operations, deployment, and troubleshooting, it is important that traceability of the requests received by I2RS agent's and actions taken be supported via a common data model.

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 datastore. This data is retained by the agent until the state is removed by the client, it is overridden by some other operation such as CLI, or the device reboots. Meaningful logging of the application and removal of changes are recommended. I2RS-applied changes to the routing element state will not be retained across routing element reboot. The I2RS datastore 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

It is expected that an I2RS agent may fail independently of the associated routing element. This could happen because I2RS is disabled on the routing element or because the I2RS agent, which may be a separate process or even running on a separate processor, experiences an unexpected failure. Just as routing state learned from a failed source is removed, the ephemeral I2RS state will usually be removed shortly after the failure is detected or as part of a graceful shutdown process. To handle these two types of failures, the I2RS agent MUST support two different notifications: a notification for the I2RS agent terminating gracefully, and a notification for the I2RS agent starting up after an unexpected failure. The two notifications are described below followed by a description of their use in unexpected failures and graceful shutdowns.

NOTIFICATION_I2RS_AGENT_TERMINATING: This notification reports that the associated I2RS agent is shutting down gracefully and that I2RS ephemeral state will be removed. It can optionally include a timestamp indicating when the I2RS agent will shut down. Use of this timestamp assumes that time synchronization has been done, and the timestamp should not have granularity finer than one second because better accuracy of shutdown time is not guaranteed.

NOTIFICATION_I2RS_AGENT_STARTING: This notification signals to the I2RS client(s) that the associated I2RS agent has started. It 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. (Note: This notification will be sent by the I2RS agent to I2RS clients that are known by the I2RS agent after a reboot. How the I2RS agent retains the knowledge of these I2RS clients is out of scope of this architecture.)

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. Since I2RS does not require a persistent connection between the I2RS client and I2RS agent, it is necessary to have a mechanism for the I2RS agent to notify I2RS clients that had subscriptions or written ephemeral state; such I2RS clients should be cached by the I2RS agent's system in persistent storage. When the I2RS agent starts, it should send a NOTIFICATION_I2RS_AGENT_STARTING to each cached I2RS client.

Graceful shutdowns: In this case, the I2RS agent can do specific limited work as part of the process of being disabled. The I2RS agent must send a NOTIFICATION_I2RS_AGENT_TERMINATING to all its cached I2RS clients. If the I2RS agent restarts after a graceful termination, it will send a NOTIFICATION_I2RS_AGENT_STARTING to each cached I2RS client.

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 client-agent interaction; 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. (For I2RS state in the presence of two I2RS clients, please see Sections 1.2 and 7.8)

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 Configuration

Changes may originate from either Local Configuration 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. The deterministic manner is the result of general I2RS rules, system rules, knobs adjusted by operator-applied policy, and the rules associated with the YANG data model (often in "MUST" and "WHEN" clauses for dependencies).

The operator-applied policy knobs can determine whether the Local Configuration overrides a particular I2RS client’s request or vice versa. Normally, most devices will have an operator-applied policy that will prioritize the I2RS client’s ephemeral configuration changes so that ephemeral data overrides the Local Configuration.

These operator-applied policy knobs can be implemented in many ways. One way is for the routing element to configure a priority on the Local Configuration and a priority on the I2RS client’s write of the ephemeral configuration. The I2RS mechanism would compare the I2RS client’s priority to write with that priority assigned to the Local Configuration in order to determine whether Local Configuration or I2RS client’s write of ephemeral data wins.

To make sure the I2RS client’s requests are what the operator desires, the I2RS data modules have a general rule that, by default, the Local Configuration always wins over the I2RS ephemeral configuration.

The reason for this general rule is if there is no operator-applied policy to turn on I2RS ephemeral overwrites of Local Configuration, then the I2RS overwrites should not occur. This general rule allows the I2RS agents to be installed in routing systems and the communication tested between I2RS clients and I2RS agents without the I2RS agent overwriting configuration state. For more details, see the examples below.

In the case when the I2RS ephemeral state always wins for a data model, if there is an I2RS ephemeral state value, it is installed instead of the Local Configuration state value. The Local Configuration information is stored so that if/when an I2RS client removes I2RS ephemeral state, the Local Configuration state can be restored.

When the Local Configuration always wins, some communication between that subsystem and the I2RS agent is still necessary. As an I2RS agent connects to the routing subsystem, the I2RS agent must also communicate with the Local Configuration to exchange model information so the I2RS agent knows 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 clients; 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.

6.3.1. Examples of Local Configuration vs. I2RS Ephemeral Configuration

A set of examples is useful in order to illustrated these architecture principles. Assume there are three routers: Router A, Router B, and Router C. There are two operator-applied policy knobs that these three routers must have regarding ephemeral state.

- Policy Knob 1: Ephemeral configuration overwrites Local Configuration.
- Policy Knob 2: Update of Local Configuration value supersedes and overwrites the ephemeral configuration.

For Policy Knob 1, the routers with an I2RS agent receiving a write for an ephemeral entry in a data model must consider the following:

1. Does the operator policy allow the ephemeral configuration changes to have priority over existing Local Configuration?

2. Does the YANG data model have any rules associated with the ephemeral configuration (such as the "MUST" or "WHEN" rule)?

For this example, there is no "MUST" or "WHEN" rule in the data being written.

The policy settings are:

	Policy Knob 1	Policy Knob 2
Router A	ephemeral has priority	ephemeral has priority
Router B	Local Configuration has priority	Local Configuration has priority
Router C	ephemeral has priority	Local Configuration has priority

Router A has the normal operator policy in Policy Knob 1 and Policy Knob 2 that prioritizes ephemeral configuration over Local Configuration in the I2RS agent. An I2RS client sends a write to an ephemeral configuration value via an I2RS agent in Router A. The I2RS agent overwrites the configuration value in the intended configuration, and the I2RS agent returns an acknowledgement of the write. If the Local Configuration value changes, Router A stays with the ephemeral configuration written by the I2RS client.

Router B's operator has no desire to allow ephemeral writes to overwrite Local Configuration even though it has installed an I2RS agent. Router B's policy prioritizes the Local Configuration over the ephemeral write. When the I2RS agent on Router B receives a write from an I2RS client, the I2RS agent will check the operator Policy Knob 1 and return a response to the I2RS client indicating the operator policy did not allow the overwriting of the Local Configuration.

The Router B case demonstrates why the I2RS architecture sets the default to the Local Configuration wins. Since I2RS functionality is new, the operator must enable it. Otherwise, the I2RS ephemeral functionality is off. Router B's operators can install the I2RS code and test responses without engaging the I2RS overwrite function.

Router C's operator sets Policy Knob 1 for the I2RS clients to overwrite existing Local Configuration and Policy Knob 2 for the Local Configuration changes to update ephemeral state. To understand why an operator might set the policy knobs this way, consider that Router C is under the control of an operator that has a back-end system that re-writes the Local Configuration of all systems at 11 p.m. each night. Any ephemeral change to the network is only supposed to last until 11 p.m. when the next Local Configuration changes are rolled out from the back-end system. The I2RS client writes the ephemeral state during the day, and the I2RS agent on Router C updates the value. At 11 p.m., the back-end configuration system updates the Local Configuration via NETCONF, and the I2RS agent is notified that the Local Configuration updated this value. The I2RS agent notifies the I2RS client that the value has been overwritten by the Local Configuration. The I2RS client in this use case is a part of an application that tracks any ephemeral state changes to make sure all ephemeral changes are included in the next configuration run.

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. I2RS services should include a capability model so that peers can determine 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.

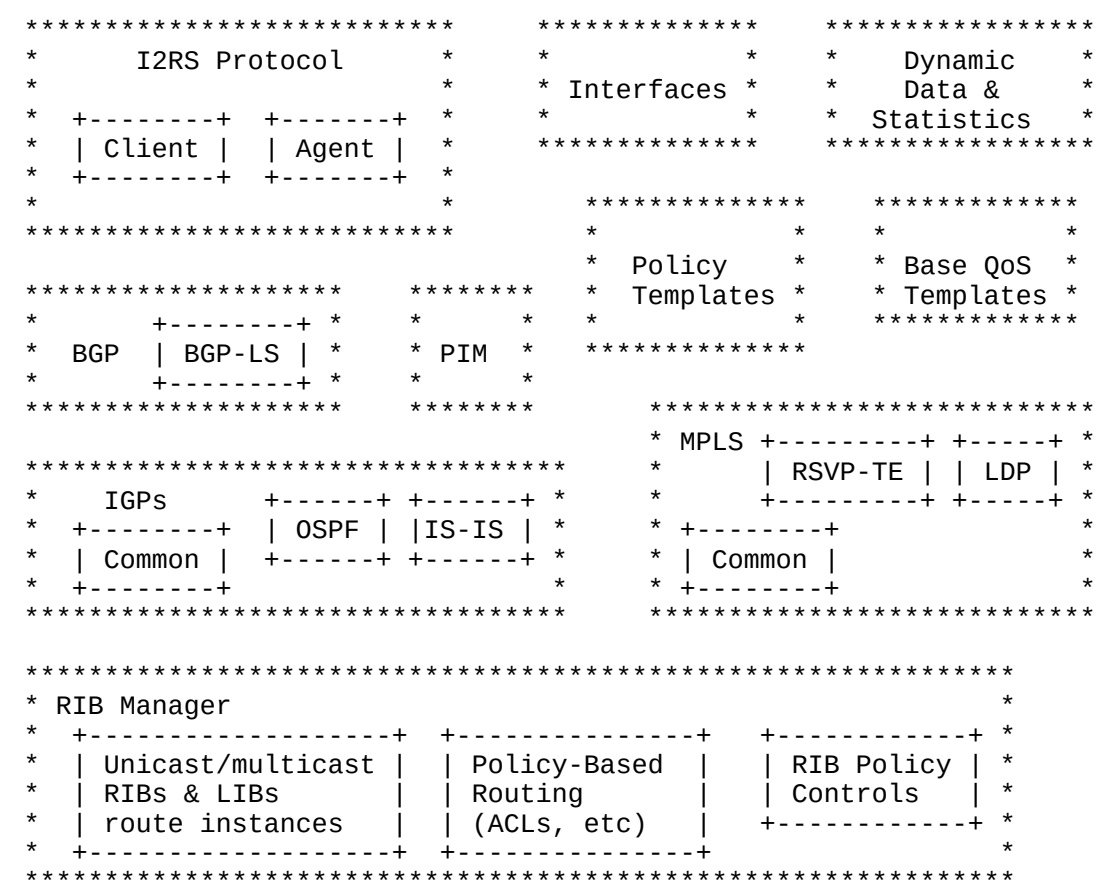

Figure 2: Anticipated I2RS Services

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 directed into). Section 6.4.5 discusses 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, per interface, or per context. 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. 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 the link-state database must not be 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 that 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 subclassing, 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 information 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. I2RS clients that 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 subclass(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 can a subclass 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.

6.4.5.2. 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, each information model should provide information regarding the following questions:

- Is X required for the data field to be accepted and applied?
- If X is optional, then how does "X" as an optional portion of the data field interact with the required aspects of the data field?
- Does the data field have defaults for the mandatory portion of the field and the optional portions of the field?
- Is X required to be within a particular set of values (e.g., range, length of strings)?

The information model needs to be clear about what read or write values are set by the client and what responses or actions are required by the agent. It is important to indicate what is required or optional in client values and agent responses/actions.

6.4.5.3. 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 allow some of the defined values to be overwritten.

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 that 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.

6.4.5.4. 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 sections list the information relationships that the information models need to support.

6.4.5.4.1. Initialization

The simplest relationship is that one object instance 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.

6.4.5.4.2. 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.

6.4.5.4.3. 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.

6.4.5.4.4. Active References

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 (maximum transmit unit), 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 assumes a data-model-driven protocol where the data models are defined in YANG 1.1 [YANG1.1] and associated YANG based model documents [RFC6991], [RFC7223], [RFC7224], [RFC7277], [RFC7317]. Two of the protocols to be expanded to support the I2RS protocol are NETCONF [RFC6241] and RESTCONF [RESTCONF]. This helps meet the goal of simplicity and thereby enhances deployability. The I2RS protocol may need to use several underlying transports (TCP, SCTP (Stream Control Transport Protocol), DCCP (Datagram Congestion Control Protocol)), 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. The transports chosen should be operator and implementor friendly to ease adoption.

Each version of the I2RS protocol will specify the following: a) which transports may be used by the I2RS protocol, b) which transports are mandatory to implement, and c) which transports are optional to implement.

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 subcomponents of a routing element and/or to multiple parts of an I2RS client. All such communication channels will use the same higher-layer I2RS protocol (which combines secure transport and I2RS contextual information). The use of additional channels for communication will be coordinated between the I2RS client and the I2RS agent using this protocol.

I2RS protocol communication may be delivered in-band via the routing system's data plane. I2RS protocol communication might be delivered out-of-band via a management interface. Depending on what operations are requested, it is possible for the I2RS protocol communication to cause the in-band communication channels to stop working; this could cause the I2RS agent to become unreachable across that communication channel.

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 that 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. Scope Policy Specifications

As Sections 4.1 and 4.2 describe, each I2RS client will have a unique identity and may have a secondary identity (see Section 2) to aid in troubleshooting. As Section 4 indicates, all authentication and authorization mechanisms are based on the primary identity, which links to a role with scope policy for reading data, for writing data, and for limiting the resources that can be consumed. The specifications for data scope policy (for read, write, or resources consumption) need to specify the data being controlled by the policy, and acceptable ranges of values for the data.

7.5. Connectivity

An I2RS client may or may not maintain an active communication channel with an I2RS agent. Therefore, an I2RS 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 I2RS client is non-functional. When communication is required, the I2RS agent or I2RS client can open a new communication channel.

State held by an I2RS agent that is owned by an I2RS 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 mechanisms 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 cleanup 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 refer to changes that 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 that 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 confidentiality, an I2RS client needs to be able to register for just the events it is interested in. It is also possible that I2RS 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 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 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 ensuring predictability is to have a simple priority associated with each I2RS client, and the highest priority change remains in effect. In the case of priority ties, the first I2RS 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 necessary that an I2RS client can register to receive notifications about changes made to writeable data, whose state is of specific interest to that I2RS client. 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 methods of error handling for a given message with multiple operations that it sends to an I2RS agent:

Perform all or none: This traditional SNMP semantic indicates that the 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 that 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 when the I2RS 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. Operational and Manageability Considerations

In order to facilitate troubleshooting of routing elements implementing I2RS agents, the routing elements should provide for a mechanism to show actively provisioned I2RS state and other I2RS agent internal information. Note that this information may contain highly sensitive material subject to the security considerations of any data models implemented by that agent and thus must be protected according to those considerations. Preferably, this mechanism should use a different privileged means other than simply connecting as an I2RS client to learn the data. Using a different mechanism should improve traceability and failure management.

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 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.

Traceability of Interactions: The ability to trace the interactions of the requests received by the I2RS agent's and actions taken by the I2RS agents is needed so that operations can monitor I2RS agents during deployment, and troubleshoot software or network problems.

Notification Subscription Service: The ability for an I2RS client to subscribe to a notification stream pushed from the I2RS agent (rather than having I2RS client poll the I2RS agent) provides a more scalable notification handling for the I2RS agent-client interactions.

9. References

9.1. Normative References

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

[RFC7920] Atlas, A., Ed., Nadeau, T., Ed., and D. Ward, "Problem Statement for the Interface to the Routing System", RFC 7920, DOI 10.17487/RFC7920, June 2016, <http://www.rfc-editor.org/info/rfc7920>.

9.2. Informative References

[I2RS-ENV-SEC] Migault, D., Ed., Halpern, J., and S. Hares, "I2RS Environment Security Requirements", Work in Progress, draft-ietf-i2rs-security-environment-reqs-01, April 2016.

[I2RS-PROT-SEC] Hares, S., Migault, D., and J. Halpern, "I2RS Security Related Requirements", Work in Progress, draft-ietf-i2rs-protocol-security-requirements-06, May 2016.

[RESTCONF] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", Work in Progress, draft-ietf-netconf-restconf-14, June 2016.

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

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

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

[RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", RFC 7224, DOI 10.17487/RFC7224, May 2014, <http://www.rfc-editor.org/info/rfc7224>.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <http://www.rfc-editor.org/info/rfc7277>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <http://www.rfc-editor.org/info/rfc7317>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <http://www.rfc-editor.org/info/rfc7752>.

[YANG1.1] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", Work in Progress, draft-ietf-netmod-rfc6020bis-14, June 2016.

Acknowledgements

Significant portions of this draft came from "Interface to the Routing System Framework" (February 2013) and "A Policy Framework for the Interface to the Routing System" (February 2013).

The authors would like to thank Nitin Bahadur, Shane Amante, Ed Crabbe, Ken Gray, Carlos Pignataro, Wes George, Ron Bonica, Joe Clarke, Juergen Schoenwalder, Jeff Haas, Jamal Hadi Salim, Scott Brim, Thomas Narten, Dean Bogdanovic, Tom Petch, Robert Raszuk, Sriganesh Kini, John Mattsson, Nancy Cam-Winget, DaCheng Zhang, Qin Wu, Ahmed Abro, Salman Asadullah, Eric Yu, Deborah Brungard, Russ Housley, Russ White, Charlie Kaufman, Benoit Claise, Spencer Dawkins, and Stephen Farrell for their suggestions and review.

Authors' Addresses

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

Email: akatlas@juniper.net

Joel Halpern
Ericsson

Email: Joel.Halpern@ericsson.com

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
United States

Phone: +1 734-604-0332
Email: shares@ndzh.com

Dave Ward
Cisco Systems
Tasman Drive
San Jose, CA 95134
United States

Email: wardd@cisco.com

Thomas D. Nadeau
Brocade

Email: tnadeau@lucidvision.com