

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

A. Atlas  
Juniper Networks  
J. Halpern  
Ericsson  
S. Hares  
ADARA  
D. Ward  
Cisco Systems  
T. Nadeau  
Juniper Networks  
July 15, 2013

An Architecture for the Interface to the Routing System  
draft-atlas-i2rs-architecture-01

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 January 16, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
1.1. Functional Overview . . . . .	3
1.2. Architectural Overview . . . . .	4
2. Terminology . . . . .	7
3. Key Architectural Properties . . . . .	9
3.1. Simplicity . . . . .	9
3.2. Extensibility . . . . .	9
3.3. Model-Driven Programmatic Interfaces . . . . .	9
3.4. Authorization and Authentication . . . . .	10
4. Network Applications and I2RS Client . . . . .	10
4.1. Example Network Application: Topology Manager . . . . .	11
5. I2RS Agent Role and Functionality . . . . .	11
5.1. Relationship to its Routing Element . . . . .	11
5.2. State Storage . . . . .	12
5.2.1. Starting and Ending . . . . .	12
5.2.2. Reversion . . . . .	13
5.3. Interactions with Local Config . . . . .	13
5.4. Routing Components and Associated I2RS Services . . . . .	14
5.4.1. Unicast and Multicast RIB and LFIB . . . . .	14
5.4.2. IGPs, BGP and Multicast Protocols . . . . .	14
5.4.3. MPLS . . . . .	15
5.4.4. Policy and QoS Mechanisms . . . . .	15
6. I2RS Client Agent Interface . . . . .	15
6.1. Protocol Structure . . . . .	15
6.2. Channel . . . . .	15
6.3. Negotiation . . . . .	16
6.4. Identity and Security Role . . . . .	16
6.5. Connectivity . . . . .	16
6.6. Notifications . . . . .	17
6.7. Information collection . . . . .	17
6.8. Multi-Headed Control . . . . .	17
6.9. Transactions . . . . .	18
7. Manageability Considerations . . . . .	19
8. Security Considerations . . . . .	19
9. IANA Considerations . . . . .	19
10. Acknowledgements . . . . .	19
11. Informative References . . . . .	19
Authors' Addresses . . . . .	20

## 1. Introduction

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

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

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

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

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

### 1.1. Functional Overview

There are four key aspects to the I2RS. First, the interface is a programmatic interface which needs to be asynchronous and offers fast, interactive access. Second, the I2RS gives access to information and state that is not usually configurable or modeled in existing implementations or configuration protocols. Third, the I2RS gives applications the ability to learn additional, structured,

filterable information and events from the router. Fourth, the I2RS will be data-model driven to facilitate extensibility and provide standard data-models to be used by network applications.

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

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

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

The existing mechanisms, such as SNMP and NetConf, that allow state to be written and read do not meet all of the key properties given in [I-D.atlas-i2rs-problem-statement] for I2RS. The overhead of infrastructure can be quite high and many MIBs do not, in definition or practice, allow writing of state. There is also very limited capability to add new application-specific state to be distributed via the routing system.

ForCES is another data-model-driven method for writing state into a router, but its focus is on the forwarding plane. By focusing on the forwarding plane, it requires that the forwarding plane be modeled and programmable and ignores the existence and intelligence of the router OS and routing system. ForCES provides a lower-level interface than I2RS is intended to address.

## 1.2. Architectural Overview

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

between the I2RS agent and the I2RS client and the associated proper behavior of the I2RS agent and I2RS client.

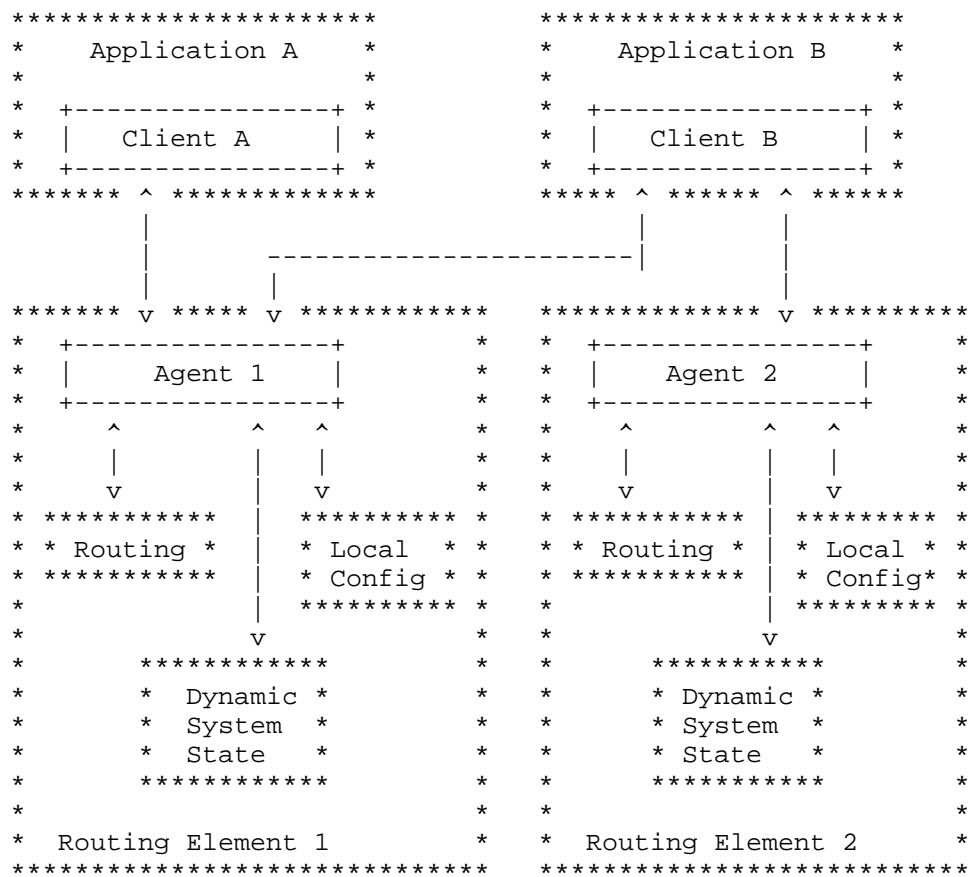


Figure 1: Architecture of I2RS clients and agents

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

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

### A server that runs BGP as a Route Reflector

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 2. Terminology

The following terminology is used in this document.

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

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

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

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

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

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

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

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

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



### 3. Key Architectural Properties

#### 3.1. Simplicity

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

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

Thus, one of the key aims for I2RS is 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?" Protocol parsimony is clearly a goal.

#### 3.2. Extensibility

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

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

#### 3.3. Model-Driven Programmatic Interfaces

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

There are several key benefits for I2RS in using model-driven architecture and protocol(s). First, it allows for transferring data-models whose content is not explicitly implemented or understood. Second, tools can automate checking and manipulating data; this is particularly valuable for both extensibility and for the ability to easily manipulate and check proprietary data-models.

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

#### 3.4. Authorization and Authentication

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

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

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

#### 4. Network Applications and I2RS Client

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

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

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

#### 4.1. Example Network Application: Topology Manager

One example of such an application is a Topology Manager. Such an application includes an I2RS client which uses the I2RS protocol to collect information about the state of the network. The Topology Manager would collect device and interface state from devices it interacts with directly. It also collects routing configuration and operation data from those devices. Most importantly, it collects information about the routing system, including the contents of the IGP (e.g. IS-IS or OSPF) and BGP data sets. This information is provided to the I2RS client using the I2RS data models and protocols.

The Topology Manager may be an integral part of an application. It would build internal data structures, and use the collected data to drive applications like path computations or anomalous routing detection. Alternatively, the Topology manager could combine the I2RS collected data with other information, abstract the result, and provide a coherent picture of the network state over another interface. That interface might use the same I2RS protocols, and could use extensions of the I2RS data models. Developing such mechanisms is outside the initial scope of the I2RS work.

### 5. I2RS Agent Role and Functionality

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

#### 5.1. Relationship to its Routing Element

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

For scalability and generality, the I2RS agent may be responsible for collecting and delivering large amounts of data from various parts of

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

## 5.2. State Storage

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

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

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

### 5.2.1. Starting and Ending

An I2RS client applies changes via the I2RS interface based on policy and other application inputs. While these changes may be of the form "do this now, and leave it there forever", they are frequently driven by other conditions which may have start times, stop times, or are only to be used under certain conditions. The I2RS interface protocol could be designed to allow an I2RS Client to provide a wide

range of such conditional information to the I2RS Agent for application. At the other extreme, the I2RS client could provide all such functionality based on its own clocking and network event reporting from the relevant I2RS Agents.

Given that the complexity of possible conditions is very large, and that some conditions may even cross network element boundaries, clearly some degree of handling must be provided on the I2RS client. As such, in this architecture it is assumed that all the complexity associated with this should be left to the I2RS client. This architectural view does mean that reliability of the communication path between the I2RS client and I2RS agent is critical. [[Editor's note: This requires more discussion in the working group.]]

#### 5.2.2. Reversion

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

#### 5.3. Interactions with Local Config

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

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

#### 5.4. Routing Components and Associated I2RS Services

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

##### 5.4.1. Unicast and Multicast RIB and LFIB

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

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

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

##### 5.4.2. IGPs, BGP and Multicast Protocols

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

- o reading the various internal rib(s) of the routing protocol is often helpful for understanding the state of the network. Directly writing these protocol-specific RIBs or databases is out of scope for I2RS.
- o reading the various pieces of policy information the particular protocol instance is using to drive its operations.
- o 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.
- o writing routes or prefixes to be advertised via the protocol.

- o joining/removing interfaces from the multicast trees

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

#### 5.4.3. MPLS

The I2RS Agent will need to interact with the knobs that policy attributes that control LDP operation as well as RSVP-TE based LSPs.

#### 5.4.4. Policy and QoS Mechanisms

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

### 6. I2RS Client Agent Interface

#### 6.1. Protocol Structure

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

#### 6.2. Channel

The uses of a single I2RS protocol does not imply that only one channel of communication is required. There may be a range of

reliability requirements, and to support the scaling there may need to be channels originating from multiple sub-components of a routing element. These will all use the date exchange protocol, and establishment of additional channels for communication will be coordinated between the I2RS client and the I2RS agent.

### 6.3. Negotiation

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

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

### 6.4. Identity and Security Role

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

### 6.5. Connectivity

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

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

There are three different assumptions that can apply to handling dead clients. The first is that the network applications or management systems will detect a dead network application and either restart that network application or clean up any state left behind. The



second is to allow state expiration, expressed as a policy associated with the I2RS client's role. The state expiration could occur after there has been no successful communication channel to or from the I2RS client for the policy-specified duration. The third is that the client could explicitly request state clean-up if a particular transport session is terminated.

#### 6.6. Notifications

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

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

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

For performance and scaling by the I2RS client and general information privacy, an I2RS Client needs to be able to register for just the events it is interested in. It is also possible that I2RS might provide a stream of notifications via a publish/subscribe mechanism that is not amenable to having the I2RS agent do the filtering.

#### 6.7. Information collection

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

#### 6.8. Multi-Headed Control

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

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

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

#### 6.9. Transactions

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

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

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

**Perform all:** In this case, the I2RS Agent will attempt to perform all the operations in the message, and will return error

indications for each one that fails. This is useful when there is no dependency across the operation, or where the client would prefer to sort out the effect of errors on its own.

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

## 7. Manageability Considerations

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

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

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

## 8. Security Considerations

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

## 9. IANA Considerations

This document includes no request to IANA.

## 10. Acknowledgements

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

The authors would like to thank Nitin Bahadur, Shane Amante, Ed Crabbe, and Ken Gray for their suggestions and review.

## 11. Informative References

[I-D.atlas-i2rs-problem-statement]

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

Authors' Addresses

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

Email: akatlas@juniper.net

Joel Halpern  
Ericsson

Email: Joel.Halpern@ericsson.com

Susan Hares  
ADARA

Email: shares@ndzh.com

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

Email: wardd@cisco.com

Thomas D. Nadeau  
Juniper Networks

Email: tnadeau@juniper.net

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

A. Atlas, Ed.  
T. Nadeau, Ed.  
Juniper Networks  
D. Ward  
Cisco Systems  
July 15, 2013

Interface to the Routing System Problem Statement  
draft-atlas-i2rs-problem-statement-01

Abstract

As modern networks grow in scale and complexity, the need for rapid and dynamic control increases. With scale, the need to automate even the simplest operations is important, but even more critical is the ability to quickly interact with more complex operations such as policy-based controls.

In order to enable applications to have access to and control over information in the Internet's routing system, we need a publicly documented interface specification. The interface needs to support real-time, asynchronous interactions using data models and encodings that are efficient and potentially different from those available today. Furthermore, the interface must be tailored to support a variety of use cases.

This document expands upon these statements of requirements to provide a detailed problem statement for an Interface to the Internet Routing System (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 January 16, 2014.

## Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. I2RS Model and Problem Area for The IETF . . . . .	3
3. Standard Data-Models of Routing State for Installation . . . . .	5
4. Learning Router Information . . . . .	5
5. Desired Aspects of a Protocol for I2RS . . . . .	6
6. Acknowledgements . . . . .	7
7. IANA Considerations . . . . .	8
8. Security Considerations . . . . .	8
9. Informative References . . . . .	8
Appendix A. Existing Management Interfaces . . . . .	8
Authors' Addresses . . . . .	9

## 1. Introduction

As modern networks grow in scale and complexity, the need for rapid, flexible and dynamic control increases. With scale, the need to automate even the simplest operation is important, but even more critical is the ability for network operators to quickly interact with these operations using mechanisms such as policy-based controls.

With complexity comes the need for more sophisticated automated applications and orchestration software that can process large quantities of data, run complex algorithms, and adjust the routing state as required in order to support the network applications, their computations and their policies. Changes made to the routing state of a network by external applications must be verifiable by those applications to ensure that the correct state has been installed in the correct places.

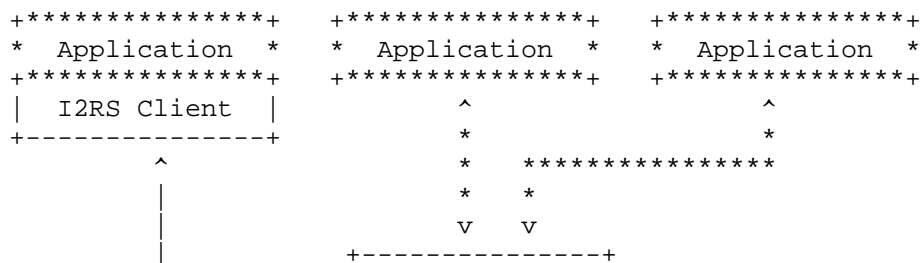
In the past, mechanisms to support the requirements outlined above have been developed piecemeal as proprietary solutions to specific

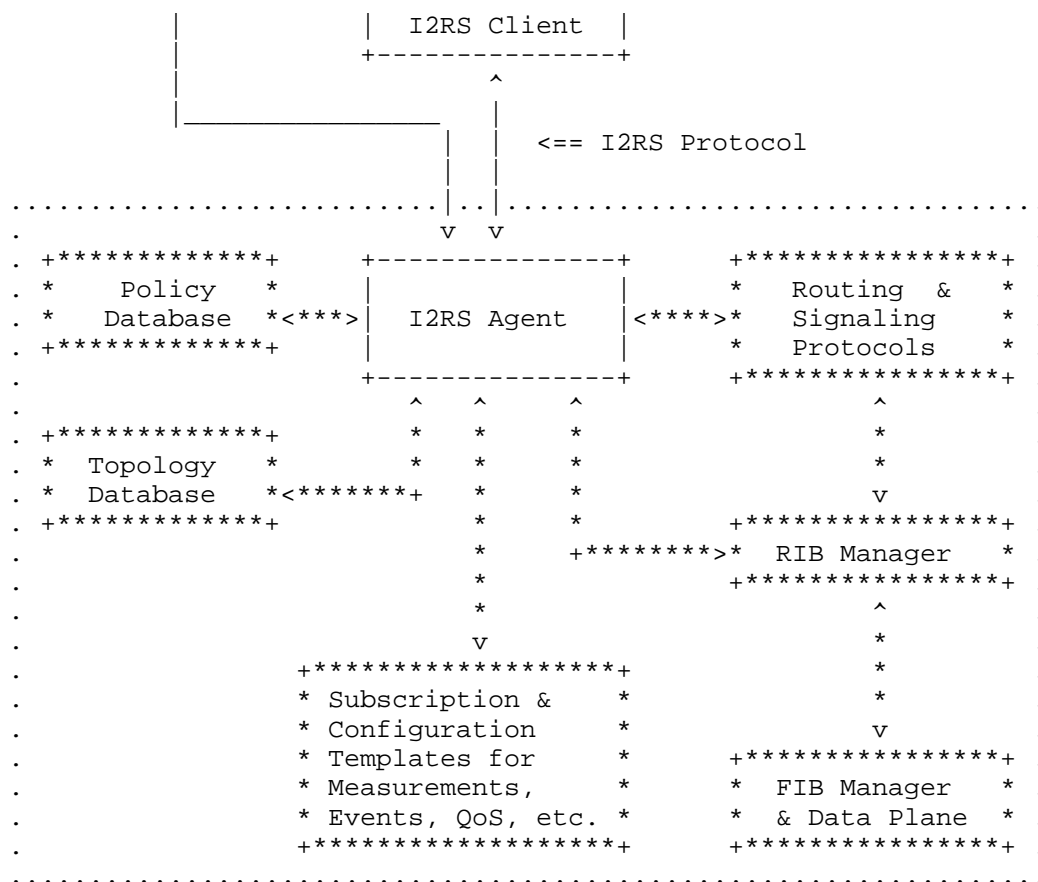
situations and needs. Many routing elements have an external interface to interact with routing - but since these vary between vendors, it is difficult to integrate use of those interfaces into a network. The existence of such proprietary interfaces demonstrates both that the need for such an interface is understood and that technological solutions are understood. What is needed are technological solutions with clearly defined operations that an application can initiate, and data-models to support such actions. These would facilitate wide-scale deployment of interoperable applications and routing systems. These solutions must be designed to facilitate rapid, isolated, secure, and dynamic changes to a device's routing system. In order to address these needs, the creation of an Interface to The Routing System (I2RS) is needed.

It should be noted that during the course of this document, we will discuss and use the term "applications". This is meant to refer to an executable program of some sort that has access to a network, such as an IP network.

## 2. I2RS Model and Problem Area for The IETF

Managing a network of production devices running a variety of routing protocols involves interactions with an between multiple components within a device. Some of these components are virtual while some are physical; it may be desirable for many, or even all of these components to be made available to be managed and manipulated by applications, given that appropriate access, authentication, and policy hurdles have been crossed. The management of only some of these components require standardization, as others have already been standardized. The I2RS model is intended to incorporate existing mechanisms where appropriate, and to build extensions and new protocols where needed. The I2RS model and problem area for IETF work is illustrated in Figure 1. The I2RS Agent is associated with a routing element, which may or may not be co-located with a data-plane. The I2RS Client is used and controlled by one or more network applications; they may be co-located or the I2RS Client might be part of a separate application, such as an orchestrator or controller.





<--> interfaces inside the scope of I2RS

+--+ objects inside the scope of I2RS

<\*> interfaces NOT within the scope of I2RS

+\*\*+ objects NOT within the scope of I2RS

.... boundary of a router participating in the I2RS

Figure 1: I2RS model and Problem Area

A critical aspect of I2RS is defining a suitable protocol or protocols to carry messages between the I2RS Clients and the I2RS Agent, and defining the data-models for use with those I2RS protocol(s). The data models should translate into a clear transfer syntax that is straightforward for applications to use (e.g., a Web



Services design paradigm), and should provide the key features specified in Section 5. The information should use existing transport protocols to provide the reliability, security, and timeliness appropriate for the particular data.

The second critical aspect are semantic-aware data-models for information in the routing system and in a topology database. The data-model should describe the meaning and relationships of the modeled items. The data-models should be separable across different features of the managed components, versioned, and extendable. An application should be able to combine data from individual routing elements to provide network-wide data-model(s).

### 3. Standard Data-Models of Routing State for Installation

There is a need to be able to precisely control routing and signaling state based upon policy or external measures. This can range from simple static routes to policy-based routing to static multicast replication and routing state. This means that, to usefully model next-hops, the data model employed needs to handle next-hop indirection (e.g. a prefix X is routed like prefix Y) as well as different types of tunneling and encapsulation. The relevant MIB modules (for example [RFC4292]) lack the necessary generality and flexibility. In addition, by having I2RS focus initially on interfaces to the RIB layer (e.g. RIB, LIB, multicast RIB, policy-based routing), the ability to use routing indirection allows flexibility and functionality that can't be as easily obtained at the forwarding layer.

Efforts to provide this level of control have focused on standardizing data models that describe the forwarding plane (e.g. ForCES [RFC3746]). I2RS posits that the routing system and a router's OS provide useful mechanisms that applications could usefully harness to accomplish application-level goals.

In addition to interfaces to the RIB layer, there is a need to configure the various routing and signaling protocols with differing dynamic state based upon application-level policy decisions. The range desired is not available via MIBs at the present time.

### 4. Learning Router Information

A router has information that applications may require so that they can understand the network, verify that programmed state is installed in the forwarding plane, measure the behavior of various flows, and understand the existing configuration and state of the router. I2RS provides a framework so that applications can register for asynchronous notifications and can make specific requests for information.

Although there are efforts to extend the topological information available, even the best of these (e.g., BGP-LS [I-D.gredler-idr-ls-distribution]) still provide only the current active state as seen at the IGP layer and above. Detailed topological state that provides more information than the current functional status is needed by applications; only the active paths or links are known versus those potentially available (e.g. administratively down) or unknown (e.g. to peers or customers) to the routing topology.

For applications to have a feedback loop that includes awareness of the relevant traffic, an application must be able to request the measurement and timely, scalable reporting of data. While a mechanism such as IPFIX [RFC5470] may be the facilitator for delivering the data, the need for an application to be able to dynamically request that measurements be taken and data delivered is critical.

There are a wide range of events that applications could use for either verification of router state before other network state is changed (e.g. that a route has been installed), to act upon changes to relevant routes by others, or upon router events (e.g. link up/down). While a few of these (e.g. link up/down) may be available via MIB Notifications today, the full range is not - nor is there the standardized ability to set up the router to trigger different actions upon an event's occurrence so that a rapid reaction can be accomplished.

## 5. Desired Aspects of a Protocol for I2RS

This section describes required aspects of a protocol that could support I2RS. Whether such a protocol is built upon extending existing mechanisms or requires a new mechanism requires further investigation.

The key aspects needed in an interface to the routing system are:

Multiple Simultaneous Asynchronous Operations: A single application should be able to send multiple operations to I2RS without being required to wait for each to complete before sending the next.

**Very Fine Granularity of Data Locking for Writing:** When an I2RS operation is processed, it is required that the data locked for writing is very granular (e.g. a particular prefix and route) rather than extremely coarse, as is done for writing configuration. This should improve the number of concurrent I2RS operations that are feasible and reduce blocking delays.

**Multi-Headed Control:** Multiple applications may communicate to the same I2RS agent in a minimally coordinated fashion. It is necessary that the I2RS agent can handle multiple requests in a well-known policy-based fashion. Data written can be owned by different I2RS clients.

**Duplex:** Communications can be established by either the I2RS client (i.e.: that resides within the application or is used by it to communicate with the I2RS server), or the I2RS server. Similarly, events, acknowledgements, failures, operations, etc. can be sent at any time by both the router and the application. The I2RS is not a pure pull-model where only the application queries to pull responses.

**High-Throughput:** At a minimum, the I2RS Agent and associated router should be able to handle a considerable number of operations per second above what basic Netconf or a proprietary CLI can.

**Responsive:** It should be possible to complete simple operations within a sub-second time-scale.

**Multi-Channel:** It should be possible for information to be communicated via the interface from different components in the router without requiring going through a single channel. For example, for scaling, some exported data or events may be better sent directly from the forwarding plane, while other interactions may come from the control-plane. Thus a single TCP session would not be a good match.

**Scalable, Filterable Information Access:** To extract information in a scalable fashion that is more easily used by applications, the ability to specify filtering constructs in an operation requesting data or requesting an asynchronous notification is very valuable.

Any ability to manipulate routing state must be subject to authentication and authorization.

## 6. Acknowledgements

The authors would like to thank Ken Gray, Ed Crabbe and Nic Leymann for their suggestions and review.

## 7. IANA Considerations

This document includes no request to IANA.

## 8. Security Considerations

Security is a key aspect of any protocol that allows state installation and extracting of detailed router state. More investigation remains to fully define the security requirements, such as authorization and authentication levels.

## 9. Informative References

- [I-D.gredler-idr-ls-distribution]  
Gredler, H., Medved, J., Previdi, S., and A. Farrel,  
"North-Bound Distribution of Link-State and TE Information  
using BGP", draft-gredler-idr-ls-distribution-02 (work in  
progress), July 2012.
- [RFC3746] Yang, L., Dantu, R., Anderson, T., and R. Gopal,  
"Forwarding and Control Element Separation (ForCES)  
Framework", RFC 3746, April 2004.
- [RFC4292] Haberman, B., "IP Forwarding Table MIB", RFC 4292, April  
2006.
- [RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek,  
"Architecture for IP Flow Information Export", RFC 5470,  
March 2009.

## Appendix A. Existing Management Interfaces

This section discusses as a single entity the combination of the abstract data models, their representation in a data language, and the transfer protocol commonly used with them. While other combinations of these existing standard technologies are possible, the ways described are those that have significant deployment.

There are three basic ways that routers are managed. The most popular is the command line interface (CLI), which allows both configuration and learning of device state. This is a proprietary interface resembling a UNIX shell that allows for very customized control and observation of a device, and, specifically of interest in this case, its routing system. Some form of this interface exists on almost every device (virtual or otherwise). Processing of information returned to the CLI (called "screen scraping") is a burdensome activity because the data is normally formatted for use by a human operator, and because the layout of the data can vary from

device to device, and between different software versions. Despite its ubiquity, this interface has never been standardized and is unlikely to ever be standardized. I2RS does not involve CLI standardization.

The second most popular interface for interrogation of a device's state, statistics, and configuration is The Simple Network Management Protocol (SNMP) and a set of relevant standards-based and proprietary Management Information Base (MIB) modules. SNMP has a strong history of being used by network managers to gather statistical and state information about devices, including their routing systems. However, SNMP is very rarely used to configure a device or any of its systems for reasons that vary depending upon the network operator. Some example reasons include complexity, the lack of desired configuration semantics (e.g., configuration "roll-back", "sandboxing" or configuration versioning), and the difficulty of using the semantics (or lack thereof) as defined in the MIB modules to configure device features. Therefore, SNMP is not considered as a candidate solution for the problems motivating I2RS.

Finally, the IETF's Network Configuration (or NetConf) protocol has made many strides at overcoming most of the limitations around configuration that were just described. However, the lack of standard data models have hampered the adoption of NetConf. Naturally, I2RS may help define needed information and data models. Additional extensions to handle multi-headed control may need to be added to NetConf and/or appropriate data models.

#### Authors' Addresses

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

Email: [akatlas@juniper.net](mailto:akatlas@juniper.net)

Thomas D. Nadeau  
Juniper Networks  
1194 N. Mathilda Ave.  
Sunnyvale, CA 94089  
USA

Email: [tnadeau@juniper.net](mailto:tnadeau@juniper.net)

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

Email: wardd@cisco.com

Internet Engineering Task Force  
I2RS working group  
Internet Draft  
Category: Informational

N. Bitar  
Verizon  
G. Heron  
L. Fang  
Cisco  
R. Krishnan  
Brocade Communications  
N. Leymann  
Deutsche Telekom  
H. Shah  
Ciena  
S. Chakrabarti  
W. Haddad  
Ericsson

Expires: January 2014

July 15, 2013

Interface to the Routing System (I2RS) for Service Chaining:  
Use Cases and Requirements

draft-bitar-i2rs-service-chaining-00

#### Abstract

Service chaining is the concept of applying an ordered set of services to a packet or a flow. Services in the chain may include network services such as load-balancing, firewalling, intrusion prevention, and routing among others. Criteria for applying a service chain to a packet or flow can be based on packet/flow attributes that span the OSI layers (e.g., physical port, Ethernet MAC header information, IP header information, Transport and application layer information). This document describes use cases and I2RS (Information to the Rousting System) requirements for the discovery and maintenance of services topology and resources. It also describes use cases and I2RS requirements for controlling the forwarding of a packet/flow along a service chain based on packet/flow attributes.

## Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January 14, 2014.

## Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Conventions used in this document

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 RFC-2119 [RFC2119].



## Table of Contents

1. Introduction.....	4
2. Abbreviations and Definitions.....	5
2.1. Abbreviations.....	5
2.2. Definitions.....	5
3. Service Chaining Use Cases and Requirements.....	5
3.1. Services topology.....	5
3.2. Monitoring Information.....	8
3.3. Traffic Redirection, Forwarding and Service Chaining.....	10
4. Service Chaining via BGP-based Redirection.....	12
5. Operational Considerations.....	12
6. IANA Considerations.....	12
7. Security Considerations.....	13
8. Acknowledgements.....	13
9. References.....	13
9.1. Normative References.....	13
9.2. Informative References.....	14
Authors' Addresses.....	14

## 1. Introduction

Several networking scenarios involve applying a set of services to a packet or flow. For instance, when a host in a protected zone initiates a session to a server outside the zone, the session may be directed to a chain of a Wide Area Network (WAN) application acceleration service, a network address and port translation (NAPT) service, and a firewall. On the server side, another set of services may also be applied. Such a sequence of services applied to a packet or flow is referred to as a service chain. Services in the chain may include deep packet inspection (DPI), load-balancing, firewalling, intrusion prevention, and routing among others.

Criteria for applying a service chain to a packet or flow can be based on packet/flow attributes that span the OSI layers. Such attributes may include the physical/virtual port on which the packet arrives, Ethernet MAC header information (e.g., VLAN ID), IP header information (e.g., source IP address), transport header information (e.g., TCP destination port number), and application layer information among others.

The transition from one service to the next in a service chain may be conditioned on the output of the current service, or may be non-conditional (pre-determined). A new mechanism, to be defined, may also enrich the packet transition in a service chain by passing service-specific information and/or information pertaining to preceding services in the chain along with the packet being processed. This type of mechanism and its influence are outside the scope of this document. In addition, this version of the document addresses the simple use case of pre-determined service chains applied to non-dropped packets with no additional information from preceding services. The service path for a packet/flow may be established via a management plane or routing, and may be enforced in the data plane via different mechanisms, as discussed in this document.

Services in a chain can be co-located on one system and/or physically separated across systems. In either case, a service may be running in its own virtualized system space or natively on the hosting system.

This document describes use cases and I2RS [i2rs-prob] requirements for the discovery and maintenance of services topology and resources. It also describes use cases and I2RS

requirements for controlling the forwarding of a packet/flow along a service chain based on packet/flow attributes.

## 2. Abbreviations and Definitions

### 2.1. Abbreviations

### 2.2. Definitions

## 3. Service Chaining Use Cases and Requirements

A service chain is an ordered set of services applied to a packet or flow. It is often the case that when a flow in a bidirectional session is assigned to a service chain, the reverse flow of the same session is required to traverse the same chain in the reverse order. Assigning a flow to a service chain is often defined at an abstract level. Mapping a service chain to a network requires knowledge of the available services, their locations and available resources so that services are properly engineered on the services infrastructure. This section describes requirements and applicability for such information, and for directing traffic through a service chain.

### 3.1. Services topology

In order to establish a service chain that applies to a packet/flow, it is important to have a topology of the service nodes. A service node can be a service running natively within a system (e.g., a service card or a service engine in a router), a virtual machine (VM) hosted on a server, a VM hosted on a service engine within a system (e.g., a service card in a router), or a dedicated standalone service hardware appliance. In addition, a service node may be dedicated to a customer (e.g., an IPVPN customer), globally shared across customers or a customer set of VPNs, or available to be assigned in whole or in part to a customer or a set of customer VPNs. the terms "customer" and "tenant" are used synonymously in this document. How a service node is created is outside the scope of this document. Resources on a service node that are not assigned to a customer context (e.g., VRF) will be logically referred to as a non-assigned service node with free available resources. A service node that can be shared in a global context will be referred to as a global service node. It should be noted, that once a service node is bound to a context, then it is only available for a virtual network (VN) associated with that context.

A services topology description requires the following information:

- . Service node address: A service node must have a unique address in a service topology. A service node identifier address can be:
  - an IP address when feasible. Such a service node can be a VM, a services engine within a system, or a hardware appliance.
  - o The tuple (service node IP address, hosting system IP address). This applies when there is need to identify the system hosting the service node or when the service node IP address is only reachable within the hosting system.
  - o The tuple (Hosting system IP-address, system internal identifier for the service engine). This applies when the service engine is not IP addressable and is within a system. A potential system internal identifier for a service engine may be (system\_slot\_number.subslot\_number.engine\_number).
- . For each service node, the following information is required:
  - o Supported service type (e.g., NAT, FW). A node may support multiple service types.
  - o Number of virtual contexts that can be supported. This parameter will indicate the maximum number of contexts that can be created on the service node.
  - o Number of virtual contexts (e.g., VRFs) available.
  - o Supported context type (e.g., VRF).
  - o Customer ID if the service node is dedicated to a customer. This indicates who can use this service node.
  - o List of supported (customer ID, virtual contexts). Note that one context per customer is

a degenerate case. This will be the global context for a given customer on a service node.

- . For each service node, virtual context and service type, the following information may be specified, depending on the service resource requirement. That is, some of the information listed here may not be relevant for some services.
  - o Service bandwidth capacity
    - Supported Packet rate
    - Supported Bandwidth (e,g, kbps)
  - o IP Forwarding Information Base size per address family
  - o Routing Information Base size
  - o MAC Forwarding database size
  - o Number of 64-bit statistics counters for policy-based accounting
  - o Number of supported Access lists (ACLs) per type (e.g., number of bits per ACL, and ACL type if applicable).
  - o Number of supported flows for services that require it (e.g., Firewall, NAT, stateful load-balancing, Deep Packet Inspection (DPI)) per flow type (i.e., fields identifying a flow) or flow identification key size. For systems that allow flexible memory usage across flow types and/or key sizes, it is sufficient to track available memory allocated for flows.

In addition to the services topology, it is important to have a view of the Virtual Network (VN) topology (VNT) and access points to which a services topology applies. The topology of such a VN could be relatively static, but it may also be dynamic, especially in a cloud environment where compute, storage, applications and associated networks may be created and removed over a short time scale. The description of a VN topology encompassing the access points is important in order

to enable installation of policies for service chaining at the right access points, instantiate the services if needed, and perform the necessary monitoring as described in later sections. VN topology information requirements are described in [i2rs-topology-reqts], but they need to be augmented with the following information:

- . Access ports (systems and ports) per VN. A port may be physical or logical on a physical port.
- . Addresses reachable on an access port.

### 3.2. Monitoring Information

Service chaining requires the ability to monitor the state of each service node, including liveliness and resource utilization. If a service node failure is detected, an action may be taken to create another service node and steer traffic to it. If a service node is hitting a resource utilization threshold, traffic may be directed to other service nodes, and/or additional service nodes may be created.

The following is a set of parameters that needs be monitored per service node per virtual context, and per service type as applicable. It should be noted that some services may not require all the parameters listed here to be monitored.

- . Bandwidth utilization (e.g., kbps)
- . Packet rate utilization
- . Bandwidth utilization per CoS (e.g., kbps)
- . Packet rate utilization per Cos
- . Memory utilization and available memory
- . RIB utilization per address family
- . FIB utilization per address family
- . Flow resource utilization per flow type
- . CPU utilization as applicable
- . Available storage

The following is a set of parameters that needs to be monitored globally per physical system (e.g., host server) providing services or hosting service nodes. Note that some parameters may not be needed for some services:

- . Bandwidth utilization (e.g., in kbps)
- . Packet rate utilization
- . Bandwidth utilization per Class of Service (CoS)
- . Packet rate per CoS
- . Memory utilization and available Memory
- . RIB utilization and available RIB memory if applicable per address family
- . FIB utilization and available FIB entries if applicable per address family
- . Flow resource utilization per flow type if applicable
- . CPU utilization if applicable
- . Power utilization
- . Available storage

Such information needs to be maintained on the distributed system hosting a service node, and/or service node as applicable. In addition, a mechanism to monitor the liveliness of a service node must be available. For some use cases, liveliness and resource utilization information needs to be accessible to a management/control plane that provides for creation of service nodes and orchestration of service chains. Some of this information may also be maintained in the management/orchestration system and validated with the distributed system where the services are instantiated. For some other use cases, a service node and/or hosting system may need to be programmed to update a management system with that information periodically or when a configured high watermark or low watermark is reached for a parameter. Thus, the interface to the service nodes and/or hosting systems must provide a mechanism that enables a management/control system to pull resource utilization information from these nodes and systems, and for these nodes and system to send updates on resource utilization to a designated system.

### 3.3. Traffic Redirection, Forwarding and Service Chaining

In a service chain, it is important to be able to direct traffic from one service node to another. Some solutions may provide this capability via dynamic routing, data-plane based policy-based routing, source based routing or a combination. Traffic redirection to a service chain requires the ability to program the routing system with a classification rule that identifies a packet/flow and an associated action that directs the corresponding packet(s) to the first node in the service chain. The focus in this section is on a hop-by-hop policy-based routing (PBR) and source based service routing. At the redirection point, classification rules should support the following information that encompasses Layer1-7 information, any of which may be wild-carded or left unspecified for a particular case:

- . Port
- . VLAN/VLAN stack
- . MAC source address
- . MAC destination address
- . Host/subnet Source IP address
- . Host/subnet Destination IP address
- . IP version
- . IP protocol
- . Source port/port-range
- . Destination port/port-range
- . Optionally, application-layer information such as key words in a URI, content type or user agent

As a result of the classification, an action will need to be specified to direct the matching packet to a service node, or to perform other forwarding action(s). Thus, the following actions should be supported:

- . Forward to a specified Outgoing port (physical or logical):



- o VLAN ID
  - o IP/GRE tunnel
  - o RSVP-TE tunnel
  - o Pseudowire (PW)
  - o Other types of tunneling protocols
- 
- . Steer the packet to a VRF
  - . Mirror packet to an IP destination
  - . Lookup up in the FIB. This could be the default behavior at the tail end of a chain or the result of no match.
  - . Forward the packet to a specific system that is multiple IP hops away (Layer 3 PBR). The destination system IP address must be specified along with the tunneling type. The action must result in encapsulating the packet to the destination. At the destination, a policy must be installed to apply a service in a specific context to the arriving packet, or direct the traffic to a local service node.
  - . Insert a source route header in the transmitted packet that identifies the nodes along the service path. The service route may be composed of IPv4 routes, IPv6 routes and/or a stack of MPLS labels. The source route may capitalize on existing mechanism or new mechanisms that are outside the scope of this document. At the destination, a policy must be installed to apply a service in a specific context to the arriving packet, or direct the traffic to a local service node.
  - . Insert a source route+service header that identifies the service path and the service type to be applied at each node. This will require the definition of a new header that carries such information.

The number of classification rules and associated actions, as well as the rate of programmability/removal of these rules will be highly application dependent. When the service chain is based on static policy (e.g., applied to

a port, a source subnet, a VN), these rules will be programmed on a system at the rate of provisioning. When the attributes of the policies are relatively static (e.g., applied to a fixed port in fixed wireline access), the rate of provisioning on the forwarding system could be low, on the order of few hundred per day. When the attributes are more dynamic, such as in a mobile environment on a system handling a large number of users, that rate could be much higher. In a cloud environment where tenant systems may be spun up and removed on a relatively short time scale this rate could be on the order of few hundreds to thousands a minute at a DC GW for instance. In all cases, if the state is not kept in a persistent storage on the forwarding system(s), system reboot actions will trigger the need for a high provisioning rate, on the order at few thousands per second. When policies are triggered by data-plane, the rate of policy provisioning will be on the order of flow rates and removal will be dependent on the flow duration. These rates will be highly dependent on the applications as well, but at a system that is handling a large number of flows, the protocol used in provisioning must be very efficient to handle a very large number of flows.

#### 4. Service Chaining via BGP-based Redirection

BGP-based steering of a traffic flow to a first service point may be required in certain cases. In this case, a router hosting a service node or connected to a service node will advertise a flow specification that causes a system that receives the advertisement to redirect a packet or mirror a copy of the packet that matches the flow specification to the advertising route [BGP-flowspec]. An I2RS interface to the advertising system from a control plane can help provisioning the advertising router with the appropriate BGP policy as well as install on that router a forwarding policy that directs the packet when received to the appropriate service node. Such BGP advertisements can be chained to effect the chaining of multiple services.

#### 5. Operational Considerations

#### 6. IANA Considerations

There is no IANA action required by this document.

## 7. Security Considerations

Service chaining imposes several security issues that must be addressed. First, the control system that installs policies in the forwarding plane must be trusted by the forwarding plane entity. An untrusted control system may install policies that hijack traffic, cause denial of service, or mirror traffic to an untrusted entity for eavesdropping. Thus, the communication channel between a control system and a forwarding plane entity must be authenticated, and may be encrypted. In addition, when services are being offered to multiple VPN customers with overlapping IP addresses, it is important that the customer privacy is maintained when applying a service chain to a customer packet/flow. Thus, the ability to identify the context in which a service needs to be applied is important. In addition, policies must be installed in the appropriate context. Finally, congesting a service node can result in packet drops that effectively may result in a denial of service. Thus, obtaining information about the performance of a service node is important to detect overload conditions and take corrective action.

## 8. Acknowledgements

The authors are thankful to David Allan for his valuable input and comments.

## 9. References

### 9.1. Normative References

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

[i2rs-topology-reqts] Medved, J., et al., "Topology API Requirements", draft-medved-i2rs-topology-requirements-00, February 2013. Work in progress.

[BGP-flowspec] Uttaro, J., et al., "BGP Flow-Spec Extended Community for Traffic Redirect to IP Next Hop", draft-simpson-idr-flowspec-redirect-02, November 2012. Work in progress.

## 9.2. Informative References

## Authors' Addresses

Nabil Bitar  
Verizon  
60 Sylvan Rd.  
Waltham, MA 02145  
EMail: nabil.n.bitar@verizon.com

Giles Heron  
Cisco Systems  
EMail: giheron@cisco.com

Luyuan Fang  
Cisco Systems  
111 Wood Avenue South  
Iselin, NJ 08830  
EMail: lufang@cisco.com

Ram Krishnan  
Brocade Communications  
San Jose, CA 95134  
EMail: ramk@brocade.com

Nicolai Leymann  
Deutsche Telekom  
Winterfeldtstrasse 21-27  
10781 Berlin  
Germany  
EMail: n.leymann@telekom.de

Himanshu Shah  
Ciena  
EMail: hshah@ciena.com

Samita Chakrabarti  
Ericsson  
EMail: samita.chakrabarti@ericsson.com

Internet-Draft

I2RS for Service Chaining

July 2013

Wassim Haddad

Ericsson

EMail: [wassim.haddad@ericsson.com](mailto:wassim.haddad@ericsson.com)

Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: January 12, 2014

K. Beck  
N. Cam-Winget  
D. McGrew  
Cisco Systems  
July 11, 2013

Using the Publish-Subscribe Model in the Interface to the Routing System  
draft-camwinget-i2rs-pubsub-sec-00

## Abstract

In the Publish-Subscribe model, subscribers express their interest in an event, or a pattern of events, and are subsequently notified of any event generated by a publisher that matches their registered interest. The model is well suited for communication in large-scale and loosely coupled distributed systems. This document describes how the model fits into Interface to the Routing System (I2RS) and Software Defined Networking (SDN) architectures, and analyzes its advantages, its security requirements, and its use in providing security within 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 January 12, 2014.

## Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

## Table of Contents

1. Introduction . . . . .	4
2. Publish-Subscribe Models . . . . .	4
2.1. Terminology . . . . .	4
3. An I2RS Publish-Subscribe Model . . . . .	5
3.1. Role of the Message Broker . . . . .	6
4. Proposed Taxonomy based on Use Cases . . . . .	6
5. Security Requirements . . . . .	11
6. Security Considerations . . . . .	14
7. IANA Considerations . . . . .	15
8. Acknowledgements . . . . .	15
9. References . . . . .	15
9.1. Normative References . . . . .	15
9.2. Informative References . . . . .	15
Authors' Addresses . . . . .	16



## 1. Introduction

This document describes the use of a publish-subscribe (or pub-sub) model to facilitate communication and authorized access for the different types of programmatic interfaces and types of applications that may be available in the I2RS and SDN architectures. It also analyzes the advantages in both scalability and security in its use within I2RS. The security requirements of a pub-sub system are given special attention, to ensure that the system meets the requirements when it is used to provide security.

## 2. Publish-Subscribe Models

There are two classical publish-subscribe models: topic- or content-based [manyfaces][pub.sub.evolution]. In a topic-based model, information is exchanged through a set of predefined "topics" or subjects; where a publisher is responsible for defining the classes of information to which a subscriber registers. In a content-based model, information is only shared to a subscriber if the specific information matches the subscriber's criteria.

In some important scenarios, a publish-subscribe model has significant advantages over a client-server model. When a source generates data intermittently, a client-server model can be used by having the client poll the server. But this method suffers from overhead in both processing and bandwidth, and it introduces a latency between the time the data is generated at the source, and the time that it is transported to the client. In contrast, a publish-subscribe model avoids the extra processing, bandwidth, and latency by establishing a channel by which the source asynchronously communicates its data.

### 2.1. Terminology

- o Publisher: defines an entry point or handle by which a protocol or programmatic interface and its capabilities such as its time delivery capabilities, protocol transport and security properties can be used.
- o Subscriber: defines an interested routing element or application requiring access to a protocol or programmatic interface.
- o Message Broker (MB): the authorization agent and broker used to manage the supported protocols and interfaces inclusive of their (access and transport) capabilities and manages the authorization of subscribers.

### 3. An I2RS Publish-Subscribe Model

Use cases and framework architectures for I2RS and SDN define the need for interfaces for acquiring information about the routing system as well as manipulating and controlling the topology and behavior of such routing system. The uses cases and frameworks described in [I-D.amante-i2rs-topology-use-cases] and [I-D.ward-i2rs-framework], respectively, describe the need for interfaces with varying capabilities. The characteristics of these capabilities include:

- o Time delivery sensitivity: interfaces or operations to be executed in the I2RS may come with different time constraints. Per section 3.5 of [I-D.ward-i2rs-framework], it is necessary in some cases to define when an operation is to be handled. In particular, there are operations that initially require synchronization of state.
- o Support for multiple protocols or implementation layers: it is expected that there would be more than a single mechanism defined to acquire, manipulate and control the routing system.
- o Secure, authorized communications: as the application(s) control the behavior of the routing system, the application must be authorized to manipulate and control the routing system, and that system must check that the application has the appropriate authorizations.
- o Support for a range of data delivery content: especially in interfaces where information (such as topology data or security monitoring or auditing data) is being conveyed, the size or amount of data to be transmitted can be very large. Conversely, interfaces that control routing may transmit very short packets.

Given the different characteristics and presence of multi-protocol support, a publish-subscribe model can be used as a means to facilitate secure authorized communications. Publishers can define the characteristics and capabilities supported by the particular interface through the message broker from which subscribers can register. Furthermore, as suggested by [I-D.amante-i2rs-topology-use-cases] and [I-D.atlas-i2rs-policy-framework], a "Policy manager" or "Policy Framework" is needed to ensure authorized communications. The message broker of a publish-subscribe model can behave as the authorizing agent and determine if a subscriber is authorized to register to specific subscriptions. Similarly, the message broker can also decide whether a publisher is authorized to provide the protocols and interfaces it is attempting to publish.

The use of the publish-subscribe pattern handles scalability issues especially given the many to many relationships. It is expected that an application will establish connections to more than one interface; similarly, an interface will be communicating with many applications. Given the many to many expected communications, the need to manage the connections and its security properties can be diminished through the use of the publish-subscribe model. The publish-subscribe model reduces the number of relationships to  $[P+S]$  (where  $P$  are the number of publishers and  $S$  are the number of subscribers) versus the potential for having to manage  $P*S$  relationships.

### 3.1. Role of the Message Broker

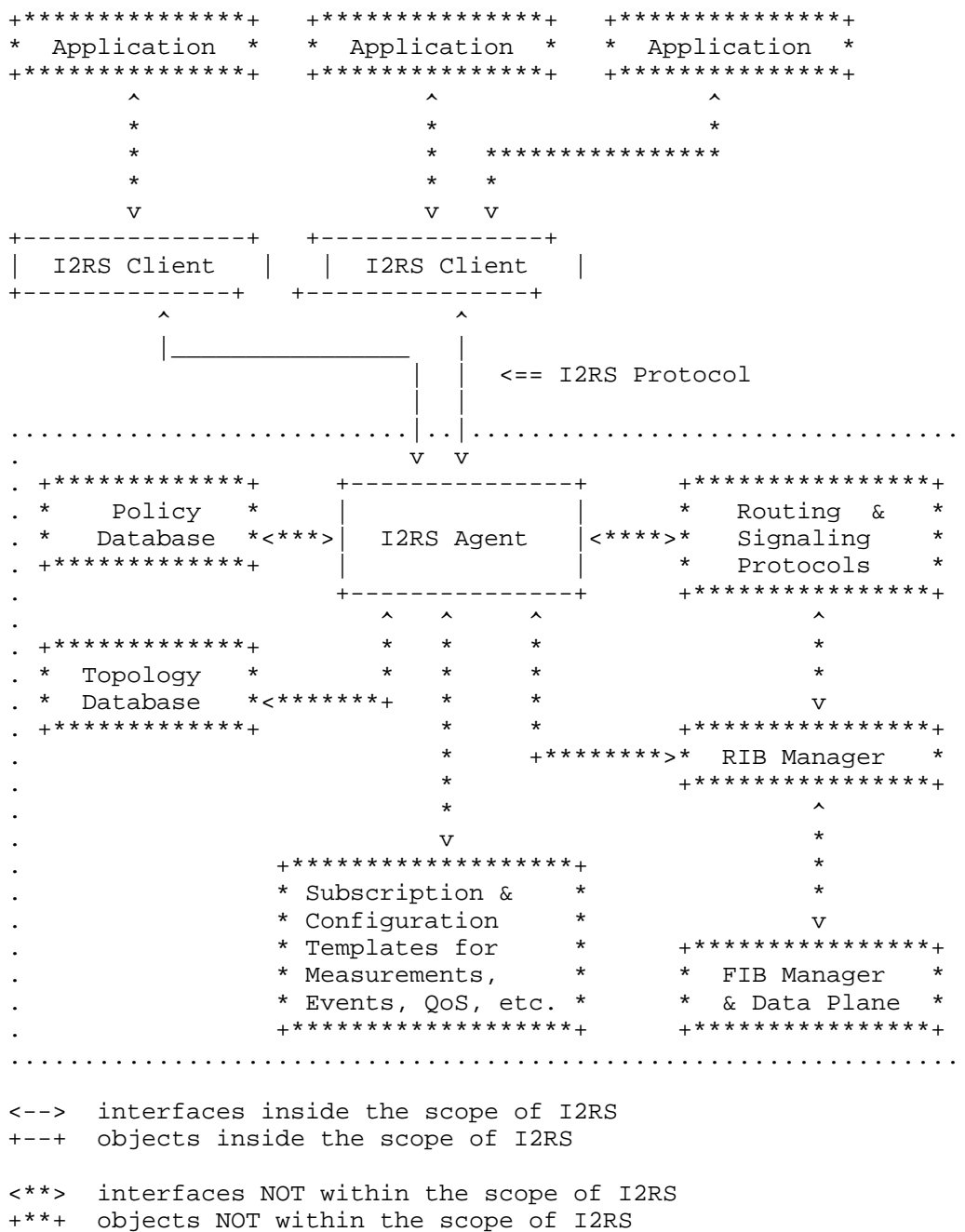
In using the publish-subscribe model, there is a Message Broker that mediates between the publishers and subscribers. The Message Broker as the intermediary, allows publishers to post their information while allowing subscribers to register to the types of information it wants to receive. As the intermediary, the Message Broker can also provide filtering capabilities to allow for a publisher to post its information once but filter according to what subscribers may be interested or is authorized to receive in a subset of what a publisher may post.

In the I2RS and Software Defined Networking (SDN) use case, the Message Broker (MB) can establish the authorizations of both publishers and subscribers. When a publisher registers with the MB, the publisher and MB authenticate each other and the MB authorizes the publisher as being authoritative for a particular topic (or if the MB is not authorized, its registration is rejected). When a subscriber registers with the MB, the subscriber and MB authenticate, and the MB authorizes the subscriber to receive the topic (or its registration is rejected).

## 4. Proposed Taxonomy based on Use Cases

Suggested architectures of the I2RS system contains I2RS Clients [I-D.amante-i2rs-topology-use-cases] [I-D.atlas-i2rs-problem-statement] (also called Commissioners [I-D.atlas-i2rs-policy-framework]) at the application level, I2RS Agents at the network level with the I2RS protocol as the interface between the two. The specific details and nature of the Clients or Commissioners and Agents require more investigation. This discussion assumes little about them and focuses on the I2RS Protocol and how the publish-subscribe model can address many of the stated requirements and use cases, and bring additional benefits such as scalability and security.

A suggestive network architecture diagram from [I-D.atlas-i2rs-problem-statement] below:



```
.... boundary of a router participating in the I2RS
```

### I2RS Architectural Model

The I2RS Agent communicates with the network platform on which it resides presumably largely with a platform specific interface, i.e. CLI or REST, and with existing protocols where appropriate. There are opportunities to facilitate the publish-subscribe model within the network platform also. Here the applicability would most likely be to new areas of functionality where no existing broadly adopted standards are used; additionally where such existing standards might be extended by adoption of publish-subscribe. It is unlikely the standards themselves would be modified but rather publish-subscribed might be layered on top to better facilitate sharing of state maintained by such standards.

As suggested, the standards used would be publish-subscribed as a new layer to improve the overall system scalability and facilitate operations such as:

- o Discovery: to address the many versions of interfaces and protocols supported, a discovery mechanism may be introduced by which I2RS Agents register as publishers or subscribers. As a publisher, an interface, schema or protocol may be "advertised" with its capabilities such as the supported versions, security and data transport properties.
- o Security: it is imperative that I2RS Agents be authenticated and authorized to employ the different interfaces and protocols. To address the many-to-many relationships, the use of a publish-subscribe model as a new layer on top helps address the security requirements in a scalable manner as well.

Taxonomy is still being developed for I2RS and there is inconsistent terminology being used to date. The terminology used here and its relationship to terminologies of others is as follows:

#### Application

I2RS application that uses the I2RS interface to interact with the routing system. This may include a Client which actually implements the application side of the I2RS interface. This has also been called the Commissioner.

#### Router

The network element that supports the I2RS interface acts as a northbound API. This may include an I2RS Agent or Server.

Areas where publish-subscribe can be deployed to satisfy stated requirements and use cases:

#### Asynchronous Router to application notifications

In the publish-subscribe model, routers register as publishers of notifications and applications interested in receiving notifications register as subscribers. The I2RS Agent in the router need only publish a notification to publish-subscribe and is unburdened from maintaining which, if any, application is interested in the notification. Similarly, the application need only express interest in receiving notifications and is unburdened from monitoring about routers. The publish-subscribe mechanism handles the asynchronous notification connecting router notifications with interested applications. Note that an application can also act as a publisher and an I2RS agent can act as a subscriber.

#### Many to Many

[I-D.amante-i2rs-topology-use-cases] and [I-D.ward-i2rs-framework] both discuss the need for the I2RS interface to support multiple applications interfacing with multiple routers and the required capability of each application to be made aware of changes made by another. With publish-subscribe routers register to publish change notifications, applications register to receive change notifications and the publish-subscribe mechanism handles the change notifications connecting router notifications with interested applications.

#### Topology

[I-D.amante-i2rs-topology-use-cases] and [I-D.ward-i2rs-framework] discuss the requirement for applications to monitor network topology and changes to the topology whether made by devices appearing or disappearing due device reboot or failure, modifications by other applications or by some other autonomic mechanism and the limitations of existing protocols to satisfy this requirement. Here, device agents, applications and other entities modifying topology would register as publishers of topology info, with publish-subscribe handling distributing change notifications to interested applications. This particular use case highlights the need for an initial synchronization to enable a subscriber to learn the current

network topology as well as an asynchronous method to learn the changes and updates to that topology as they occur.

#### RIB Updates

[I-D.ward-i2rs-framework] and others describe the need for router RIB updates to be available to I2RS applications. This is another case where routers can register as publishers of RIB changes with publish-subscribe handling the distribution of these changes to interested applications.

#### Policy Management

[I-D.atlas-i2rs-policy-framework] discusses the need for applications to monitor policy changes, including those made by other applications. This would be a subset of the Many to Many publish-subscribe case.

Other cases which it is clear would be well handled by publish-subscribed include Events and Configuration Changes. Use cases from [I-D.keyupate-i2rs-bgp-usecases] would include:

BGP Error Notifications. Notification of Routing Events.

BGP Protocol Statistics. Routing agents could publish BGP errors, other BGP events and BGP statistics on the router.

Tracing Dropped BGP Routes. Routers can publish learning of BGP routes which would enable applications the monitor the propagation of routes through the AS.

## 5. Security Requirements

This section describes security requirements as needed to sustain an I2RS or SDN. These requirements are based on the use cases defining the need for multiple protocol (using multiple layers) that need to act at different time sensitivities. It is expected that applications will need to gain appropriate authorization to use one or more of these protocols within an I2RS or SDN.

In access control models, it is common to describe access control on data in terms of the entities that are permitted to read that data, and the entities that are permitted to write that data. These models naturally apply to a publish-subscribe model: a subscriber to a topic is authorized to read data on that topic, and a publisher is authorized to write data on it. In a pub-sub model, there may be



many subscribers to a topic, and there may be more than one publisher on a topic as well.

Published data requires the following protections, which are stated in terms of a specific topic:

Authentication: it must not be possible for any entity other than the publisher to create a message that a subscriber will accept as authentic. It must not be possible for one subscriber to create messages that are accepted by the other subscribers to the same topic. When there are multiple publishers on a particular topic, it must be possible for the subscribers to authenticate the actual publisher of each message.

Anti-replay: if a subscriber receives the same exact message twice (e.g. because an attacker has copied and then re-injected the message), it must be detectable, and the subscriber must reject the replayed message.

Ordering: it must not be possible for any entity to re-order the authentic messages in such a way that the subscriber would accept the messages in a sequence other than the one intended by the publisher. If there are multiple publishers and the system does not ensure that they are delivered in a particular order, then subscribers (and the applications that use them) must not rely on any particular ordering.

Confidentiality: it should not be possible for any entity other than a subscriber to read the messages.

Authenticity, anti-replay, and ordering are required, because those protections are essential to prevent the manipulation of the routing system. Confidentiality may not always be necessary, but it is strongly recommended that it be available within the pub-sub system. Anti-replay and ordering can be easily achieved whenever authentication is available, through the use of sequence numbers and/or timestamps.

There are different cryptographic techniques that can be used to provide the security services outlined above. One method is to use pairwise communications security, such as TLS or IPsec, between each subscriber and the MB and between each publisher and the MB (in the case that all communication goes through the MB). Mutual authentication between the MB and the publishers and subscribers is required. The minimum configuration that is necessary is that each publisher, and each subscriber, be configured with the information needed to authenticate the MB. Because each communication channel is separately protected, all of the needed security services are

provided and separation is enforced between different publishers and subscribers. The advantage of this method is its simplicity. It does have disadvantages when there are many subscribers and/or publishers: cryptographic operations will need to be performed for each subscriber, and if the MB is compromised, then the security of the entire system is compromised. If the MB functionality is distributed to multiple nodes in the network, that may help scalability, but at the expense of security, since it creates additional points in the system whose compromise will undermine its security.

In some cases the communication may go directly between a publisher and a subscriber, instead of through the MB. Those cases have similar advantages and disadvantages. In these cases, the MB must provide the subscribers and publishers with the information that they need to authenticate each other, and to check the authorizations of the other side of the secure communications channel. This authentication and authorization information can be provided by the MB on a per-session basis, or it could be persistent across multiple sessions. If the data can persist for a long time, then it is important to have a method by which the MB can revoke the authorizations.

Another method is to use cryptography on the messages themselves. Digital signatures can be used to provide authentication of each message, in which each publisher has a private key, and each subscriber has the corresponding public key. Confidentiality can be provided using a group key that is shared by each publisher and the set of corresponding subscribers. This method has the advantage that cryptographic operations need only be done once on each method, thus enabling the system to scale well when there are large numbers of subscribers. It also reduces the number of keys used, and the amount of session state that needs to be maintained by the MB (or by the publishers, in the case of direct communication). The compromise of the MB does not directly compromise the entire system in this case (though if the MB is authoritative regarding which public keys should be trusted, an attacker who compromises it can always perpetrate a man-in-the-middle attack).

Security requirements using the publish-subscribe model include:

- o REQ1: Mutual authentication between the Publishers and the Message Broker, and the Subscribers and the Message Broker, is REQUIRED. Authentication to the Message Broker MUST be established as the minimum to determine authorization as either a publisher or subscriber.

- o REQ2: A Message Broker must exist to determine whether the routing element or application is authorized to access the particular protocol or interface (e.g. whether it is allowed to publish or subscribe).
- o REQ3: A Publisher SHOULD define the security properties of its protocol or program interface (e.g. how its messages are secured, using TLS for instance). Its transport SHOULD provide confidentiality and MUST provide message authentication.
- o REQ4: A Publisher SHOULD describe the latency with which it can deliver messages, and a Subscriber SHOULD verify that the latency is acceptable. This is needed to protect applications from attacks that block the timely delivery of critical information.
- o REQ5: The I2RS interface's communication channel must provide confidentiality and message authentication.
- o REQ6: When there are multiple subscribers, it should be possible to provide cryptographic authentication in such a way that no subscriber can pose as a publisher for which it subscribed.
- o REQ7: Versioning MUST be supported. Backwards compatibility of interfaces greatly simplifies the system, but cannot always be expected. Version negotiation SHOULD be provided, and can be facilitated through the publish-subscribe layer as the Message Broker must account for the existence of multiple versions of interfaces and protocols.
- o REQ8: A discovery mechanism, when used, must be secured. At a minimum, it must be possible to configure an element with information that enables it to authenticate the provider of the discovery service, or the discovered data, and reject data from untrusted sources. A discovery service SHOULD have the ability to authenticate its clients and choose to withhold information from a client based on its authorizations.

## 6. Security Considerations

As the interfaces and frameworks being defined within I2RS and SDN are purposed to inform, manipulate and control topology or behavior of a routing system they must be secured through proper authentication and authorization. Section 5 defines the security requirements to address appropriate control access, privacy and authenticity.

## 7. IANA Considerations

This memo includes no request to IANA.

## 8. Acknowledgements

The authors thank Allan Thomson and Anthony Grieco for valuable review and comments on this draft.

## 9. References

### 9.1. Normative References

- [I-D.amante-i2rs-topology-use-cases]  
Amante, S., Medved, J., Previdi, S., and T. Nadeau,  
"Topology API Use Cases",  
draft-amante-i2rs-topology-use-cases-00 (work in  
progress), February 2013.
- [I-D.atlas-i2rs-policy-framework]  
Atlas, A., Hares, S., and J. Halpern, "A Policy Framework  
for the Interface to the Routing System",  
draft-atlas-i2rs-policy-framework-00 (work in progress),  
February 2013.
- [I-D.atlas-i2rs-problem-statement]  
Atlas, A., Nadeau, T., and D. Ward, "Interface to the  
Routing System Problem Statement",  
draft-atlas-i2rs-problem-statement-00 (work in progress),  
February 2013.
- [I-D.ward-i2rs-framework]  
Atlas, A., Nadeau, T., and D. Ward, "Interface to the  
Routing System Framework", draft-ward-i2rs-framework-00  
(work in progress), February 2013.

### 9.2. Informative References

- [manyfaces]  
Eugster, P., Felber, P., Guerraoui, R., and A-M.  
Kermarrec, "The many faces of publish/subscribe",  
ACM Computing Surveys, Volume 35, Issue 2, pages 114-131,  
2003.
- [pub.sub.evolution]  
Schiper, A., "The Evolution of Publish/Subscribe

Communication Systems", Springer Volume 2584 of Lecture  
Notes in Computer Science, pages 137-141, 2003.

Authors' Addresses

Ken Beck  
Cisco Systems

Email: kebeck@cisco.com

Nancy Cam-Winget  
Cisco Systems

Email: ncamwing@cisco.com

David McGrew  
Cisco Systems

Email: mcgrew@cisco.com



Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: September 12, 2013

K.P. Patel  
R. Fernando  
Cisco Systems  
H. Gredler  
Juniper Networks  
S. Amante  
Level 3 Communications, Inc.  
March 11, 2013

Use Cases for an Interface to BGP Protocol  
draft-keyupate-i2rs-bgp-usecases-00.txt

Abstract

A network routing protocol like BGP is typically configured and results of its operation are analyzed through some form of Command Line Interface (CLI) or NETCONF. These interactions to control BGP and diagnose its operation encompass: configuration of protocol parameters, display of protocol data, setting of certain protocol state and debugging of the protocol.

Interface to the Routing System's (I2RS) Programmatic interfaces, as defined in [I-D.ward-i2rs-framework], provides an alternate way to control the configuration and diagnose the operation of the BGP protocol. I2RS may be used for the configuration, manipulation, polling or analyzing protocol data. This document describes set of use cases for which I2RS can be used for BGP protocol. It is intended to provide a base for the solution draft describing a set of interfaces to the BGP protocol.

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 September 12, 2013.

## Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

## Table of Contents

1. Introduction . . . . .	3
1.1. Requirements Language . . . . .	3
2. BGP Configuration . . . . .	3
2.1. BGP Protocol Configuration . . . . .	4
2.2. BGP Policy Configuration . . . . .	5
3. BGP Protocol Operation . . . . .	7
3.1. BGP Error Handling for Internal BGP Sessions . . . . .	7
4. BGP Route Manipulation . . . . .	8
4.1. Customized Best Path Selection Criteria . . . . .	8
4.2. Flowspec Routes . . . . .	8
4.3. Route Filter Routes for Legacy Routers . . . . .	9
4.4. Optimized Exit Control . . . . .	9
5. BGP Events . . . . .	9
5.1. Notification of Routing Events . . . . .	10
5.2. Tracing Dropped BGP Routes . . . . .	11
5.3. BGP Protocol Statistics . . . . .	12
6. Security Considerations . . . . .	13
7. Acknowledgements . . . . .	13
8. References . . . . .	13



8.1. Normative References . . . . .	13
8.2. Informative References . . . . .	14
Authors' Addresses . . . . .	14

## 1. Introduction

Typically, a network routing protocol like BGP is configured and results of its operation are analyzed through some form of Command Line Interface (CLI) or NETCONF. These interactions to control BGP and diagnose its operation encompass: configuration of protocol parameters, display of protocol data, setting of certain protocol state and debugging of the protocol.

The I2RS Framework document [I-D.ward-i2rs-framework] describes a mechanism to control network protocols like BGP using a set of programmatic interfaces. These programmatic interfaces allow one to control the BGP protocol by analyzing its operational state and routing protocol data, plus manipulating BGP's configuration to achieve various goals. The I2RS is not intended to replace any existing configuration mechanisms, (i.e.: Command Line Interface or NETCONF). Instead, I2RS is intended to augment those existing mechanisms by defining a standardized set of programmatic interfaces to enable easier configuration, interrogation and analysis of the BGP protocol.

This document describes set of use cases for which I2RS's programmatic interfaces can be used to control and analyze the operation of BGP. The use cases described in this document cover the following aspects of BGP: protocol parameter configuration, configuration of routing policies, protocol route manipulation and tracking of protocol events. The goal is to inform the community's understanding of where the I2RS BGP extensions fit within the overall I2RS architecture. It is intended to provide a basis for the solutions draft describing the set of Interfaces to the BGP protocol.

### 1.1. Requirements Language

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 RFC 2119 [RFC2119].

## 2. BGP Configuration

The configuration of BGP is arduous to establish and maintain, particularly on networks whose services have a requirement for complex routing policies. This need is magnified by the need to routinely perform changes to large numbers of BGP routers to, for example: add or remove customer's BGP sessions, announce or withdraw

(customer) IP prefixes in BGP, modify BGP policies to effect changes in Traffic Engineering, audit BGP routers to ensure they have consistent and appropriate BGP policies, etc.

There are three categories of BGP configuration:

1. Local BGP routing protocol configuration: local Autonomous System Number (ASN), BGP path selection properties of the router, injection of (aggregate) routes into BGP, etc.
2. Local BGP policies: policies designed to filter and then manipulate BGP attributes associated with BGP routes learned through BGP sessions. These policies typically live in the global configuration of a BGP router, but are applied on a per-BGP neighbor basis (or, group of BGP neighbors); and,
3. BGP neighbor sessions: remote ASN, remote IP address, address families, BGP policies to applied to routes, max-prefix limits, etc.

The sum total of BGP configuration on a BGP router is typically the largest quantify of configuration on Service Provider's BGP routers, by a fairly large margin. When that is combined with the large set of routine configuration changes, mentioned above, it should be fairly clear that systematic reading, configuration and control of BGP routers through a mechanism like I2RS would greatly benefit all operators of BGP routers.

While it may not be possible to provide programmatic APIs for esoteric vendor-specific policy configuration, it is possible to provide such API's for BGP protocol specific configuration and the more commonly used BGP routing policies.

## 2.1. BGP Protocol Configuration

Ability to enable and disable new address families within a BGP protocol for a network of BGP speaking routers is a challenge. The challenge is mainly in keeping track of BGP speaker's feature capabilities and then configuration of new address families on a multiple BGP speakers within a given network. With the necessary information, I2RS controllers allow a network operator to push configuration information for enabling and disabling of new address families on a partial or entire set of BGP speakers within a given network. This would assist in building BGP overlay networks as needed.

For VPN address families, the main challenge lies in the complex VPN configuration required to setup the control plane for Customer VPNs.

The configuration involves creating a Virtual Routing and Forwarding instance (VRF), a Route Distinguisher (RD) that ensures each customer prefixes remains unique across VPNs, and Route Targets (RT) that help ensure that the Customer prefixes are segregated appropriately so that they do not cross the VPN boundaries. I2RS would allow a network operator to push such configuration from a central location where a global VPN provisioning information could be stored. This helps avoid manual configuration of a VPN on multiple routers. Instead the configuration is controlled and pushed through a central I2RS controller using a programmatic set of APIs on targeted set of BGP speakers.

Use of I2RS controllers to announce protocol configuration information would simplify and automate configuration of BGP protocol in IBGP deployments where the protocol based policies are seldom used. To facilitate such a centralized configuration model, BGP speakers could be extended to use programmatic APIs to announce their feature capabilities as part of protocol initialization to the centralized I2RS controllers. This would assist I2RS controllers to auto-discover BGP protocol capabilities of various BGP speakers in a given network. I2RS controllers in turn would use the information towards enabling/disabling of BGP specific features on BGP speakers.

## 2.2. BGP Policy Configuration

Filtering of BGP routes is strongly recommended to control the announcements of BGP prefixes across the internet. Most providers make extensive use of BGP prefix filtering policies at the edge of their networks. The reasons for filtering BGP prefixes are:

- o Avoid Unwanted Route Announcements. Filter prefixes that MUST not be routed [RFC5735], [RFC5156]. Filter prefixes that are not allocated by Internet Routing Registries.
- o Facilitate Route Summarization. Filter prefixes beyond certain agreed prefix mask length between providers. Route Summarization helps control BGP RIB and FIB table size.
- o Defensive Security. Filter prefixes from Stub customer ASes that are not owned by the customers. Filter customer prefixes announced by other providers. This helps avoid prefix hijacking.

A set of standards-based schemas to enable configuration of Local BGP policies and BGP neighbor sessions was realized through the Routing Policy Specification Language (RPSL) [RFC2622]. The RPSL defined a standards-based schemas, or 'objects' as it called them, that defined:

- o binding of IP prefixes to (one or more) Origin AS, (route objects);
- o collections of routes (route-set objects);
- o collections of Autonomous Systems (as-set objects); and,
- o routing policy of an Autonomous System to/from its adjacent neighbor AS'es, (aut-num objects)

Each ASN is responsible for creation, modification and deletion of its RPSL objects in an Internet Routing Registry (IRR). IRR's are typically operated by Regional Internet Registries (RIR's) and a few dozen larger ISP's and independent organizations. The IRR's provide a well-known location for all organizations attached to the Internet to retrieve or update RPSL objects.

While still widely and actively used by Internet Service Providers, the prevailing belief is that the data contained in the IRR's is inaccurate, primarily due to a lack of deployed authorization method with respect to the creation or modification of RPSL objects. It should be noted that this criticism is not directed at the previously defined RPSL schemas, but rather at the data contained in RPSL schemas by end-users of the IRR system. Please refer to the IRR And Routing Policy Configuration Considerations [I-D.mcpherson-irr-routing-policy-considerations] document for a more thorough discussion of the history and present state of the IRR's.

Currently, RPSL schemas are exchanged between non-routing systems (servers) used within the IRR system. In addition, open-source and proprietary applications create or modify RPSL schemas, as necessary, to signal the announcement (or, withdrawal) of an IP prefix from an ASN or the creation (or, teardown) of a neighbor relationship between two adjacent ASN's. Most importantly, these RPSL schemas are consumed by similar applications to automatically build routing policies, (i.e.: lists of IP prefixes, corresponding Origin ASN's and/or AS\_PATH's), that then get translated to device-specific syntax (i.e.: CLI) before being pushed into individual BGP routers to effect routing policy on the network. It is common for Internet Service Providers to perform updates to these routing policies across their entire network on a daily basis.

With I2RS it would be desirable to change the last step in the above process so that BGP policies derived from RPSL schemas, and other information sources, are translated into standards-based schemas that are then pushed, or pulled, into individual BGP routers. More generally, I2RS controllers could use API's to gather information required to build various types of BGP routing policies plus the

corresponding set of Autonomous System Border Routers (ASBR's) where such policies need to be applied in the network and, finally, making those changes to individual network elements so those BGP policies take effect in the network. In doing so, a network operator now has a centralized way of building and making these policies take effect across the network in a coordinated manner.

### 3. BGP Protocol Operation

It is increasingly common for services facilitated via BGP to be subject to severe, widespread disruptions (outages), primarily due to the destructive teardown of BGP sessions as a result of receiving malformed BGP attributes. The document Operational Requirements for Enhanced Error Handling Behaviour in BGP-4

[I-D.ietf-grow-ops-reqs-for-bgp-error-handling] outlines requirements to try to minimize the scope of the impact attributed to such errors. Unfortunately, more fine-grained BGP error handling solutions, which would result in little to no impact on the operation of BGP protocol, remain elusive.

#### 3.1. BGP Error Handling for Internal BGP Sessions

It is possible that I2RS could enable enhanced error handling techniques for Internal BGP sessions. At a minimum, I2RS-capable BGP routers could signal an event such as "Malformed Attribute Received" toward an I2RS controller(s). I2RS controller(s) may already have a real-time view of BGP routes, and corresponding BGP attributes, or may dynamically interrogate BGP routers in the network to identify the present propagation scope of the BGP route(s) that are affected. Finally, the I2RS controller(s) could then signal back to BGP routers to apply a filter that would block propagation of the BGP attribute or BGP route, as necessary, in order to temporarily aid in consistency of BGP routing information across the entire network until a permanent fix can be developed and deployed within BGP routers.

I2RS would enable the global visibility and global control over the operational state of BGP, within a given Autonomous System, that is necessary to facilitate the learning of, rapid response to and more fine-grained isolation/scoping of BGP protocol events that currently cause a destructive tear-down of BGP sessions that lead to widespread disruptions of services.

#### 4. BGP Route Manipulation

Multiprotocol BGP [RFC4760] provides support to carry routing information for different BGP address families. Route manipulation is heavily done across these different address families for different reasons. BGP IPv4 and IPv6 address families use BGP Communities [RFC1997] and other IBGP and EBGP attributes to manipulate BGP routes for Traffic Engineering purpose. BGP VPN address families use Extended Communities [RFC4360] to filter unwanted BGP routes. BGP Flowspec address family [RFC5575] is used to install Flow based filters to filter unwanted data traffic. The following sub-sections describe the use of I2RS towards BGP Route Manipulation for different BGP address families.

##### 4.1. Customized Best Path Selection Criteria

The BGP customized Bestpath facilitates custom bestpath computations within a BGP speaking network. It is usually used within an IBGP network. Customized bestpaths use special extended communities known as cost communities. Cost communities carry enough information; Point of Insertion (POI) and the cost value to signal where in BGP bestpath the customize checks need to be done. Both, the traffic engineering as well as backdoor (SHAM) links use customized bestpath computation.

With I2RS, it would be possible for an I2RS controller to push routes with custom cost communities on the BGP routers for Traffic Engineering purpose. I2RS controller now can act as a central entity keeping track of all Traffic engineering data that get applied to BGP routes within an IBGP network.

##### 4.2. Flowspec Routes

The BGP flowspec address family is used to disseminate the traffic flow specification to the BGP Autonomous System Border Routers (ASBRs) and Provider Edge (PE) routers. Both, the BGP ASBRs and the PEs would translate the received BGP traffic flow specification into an Access Control List (ACL) and install it in router's forwarding path. Using such ACLs routers can now classify, shape, rate limit, filter, or redirect traffic flows.

With I2RS, it would be possible for an I2RS controller to push traffic flow specifications to the BGP ASBRs and the PE routers. I2RS controller can act as a central entity tracking all the traffic flow specifications that are installed within an IBGP network. I2RS controller could also prioritize and control the announcement of traffic flow specifications according to various ASRBs and PE router's capacity. BGP ASBRs and PE routers MAY forward traffic flow

specifications received from EBGp speakers to I2RS Controllers. This would allow I2RS controllers to centrally manage and track any externally received traffic flow specifications.

#### 4.3. Route Filter Routes for Legacy Routers

The BGP Route Filter address family is used to disseminate the Route Target filter information between VPN BGP speakers. This information is then used to build a route distribution graph that helps in limiting the propagation of VPN NLRI within a VPN network. However, it requires that all the BGP VPN routers are upgraded to support this functionality. Otherwise, the graph information is incomplete when a VPN network consists of legacy routers that participates in VPN but does not implement the BGP route filter address family.

With I2RS, it would be possible for an I2RS controller to push router filter information to BGP RR routers on behalf of all legacy routers that participates in VPN but does not support or implement the BGP route filter address family. I2RS controller can act as a central entity tracking all the configured Route Filters for legacy routers and push them on appropriate RRs who in turn would push it to ASBRs and PE routers. In this way, I2RS controllers help build an optimal route distribution graph that would assist in filtering of VPN NLRIs in a VPN network.

#### 4.4. Optimized Exit Control

Optimized Exit Control is used to provide route optimization and load distribution for multiple network connections between networks. Network operators can monitor IP traffic flows and then could define policies and rules based on traffic class performance, link bandwidth monetary costs, link load distribution, traffic types, link failures, etc.

With I2RS, it would be possible for an I2RS controller to manipulate BGP routes and its parameters that influence BGP bestpath decisions. I2RS controller could act as a central entity that would monitor and manipulate BGP routes based on central network based policies. Such routes would then be injected by a I2RS controller into the network so as to get the load distribution for multiple network connections.

### 5. BGP Events

Given the extremely large number of BGP Routes in networks, it is critical to have scalable mechanisms that can be used to monitor for events affecting routing state and, consequently, reachability. In addition, similar tools are needed in order to monitor BGP protocol statistics, which help operators and developers better understand scalability of software and hardware that BGP utilizes.

I2RS could provide a publish-subscribe capability to applications to:

- o request monitoring of BGP routes and related events; and,
- o subscribe to the I2RS controller to receive events related to BGP routes or other protocol-related events of interest.

#### 5.1. Notification of Routing Events

There are certain IP prefixes, for example those that are arbitrarily classified by a given network operator as "high visibility" by its end-users, for which immediate notification of changes in their state are extremely useful to know about. Upon notification of such events, a Network Operations Center (NOC) could respond to customer inquiries in a more timely fashion; alternatively, the NOC may decide to perform Traffic Engineering to restore service, etc.

Currently, the only way to learn of such events is for a BGP monitoring system to establish a BGP session with a multitude of BGP routers in an AS. Then, the BGP monitoring system needs to look through all BGP UPDATE's in order to identify those events that are of interest to it. Note, this doesn't account for the fact that there are several applications that might be simultaneously interested in learning of events to a given IP prefix nor the fact that some applications may want to dynamically insert or remove "IP prefixes of interest", depending on the needs of their constituent applications.

With I2RS, it is conceivable that applications could tell an I2RS controller, through a North-Bound API, their "IP prefixes" (or, AS\_PATH's, BGP communities, etc.) that are of interest. For example, a NOC application may be interested in changes to high visibility content or service-provider Web sites; alternatively, a security application may be interested in events associated with a different set of IP prefixes. The I2RS controller would then consolidate the list of IP prefixes, and associated characteristics, to be monitored and program BGP routers in an AS to observe this subset of routes for changes. Some examples of changes in routing state might include:

- o an IP prefix being announced or withdrawn



- o an IP prefix being suppressed, due to route flap dampening
- o an alternative best-path being chosen for a given IP prefix

When the requisite events for a BGP Route are observed by a BGP router, it would notify I2RS controllers.

The I2RS controllers would have a publish/subscribe mechanism whereby various sets of applications may subscribe to events of interest. The I2RS controller would then publish these events so applications would immediately receive them and take the appropriate domain-specific action necessary.

## 5.2. Tracing Dropped BGP Routes

It is extremely useful to operators to be able to rapidly identify instances where a BGP route is not being propagated within an Autonomous System. At a minimum, this could result in sub-optimal performance when attempting to reach such destinations.

There are two instances when this scenario will occur. First, when a Service Provider is using "Soft Reconfiguration Inbound", it allows their ASBR routers to receive a copy of a BGP route, but show that route was not permitted into the Adj-RIB-In most likely as a result of the inbound BGP policy not permitting that IP prefix. Thus, this BGP route is not even eligible for BGP Path Selection. The second instance is where the BGP route is permitted by the inbound BGP policy into the Adj-RIB-In, but due to BGP Path Selection (i.e.: lower LOCAL\_PREF, longer AS\_PATH length, etc.) was not chosen as the best path and, subsequently, this particular BGP route is not forwarded on to other internal BGP speakers in the AS. In both instances, the BGP route is only visible within the ASBR on which that BGP route was first learned. Needless to say, in large Service Provider networks with a numerous interconnects to a single customer it can be very time-consuming to discover where such a BGP route is learned before ultimately determining why the route was blocked or not preferred.

With I2RS, it would be possible for an I2RS controller to rapidly gather information from across a large set of BGP routers in the network to determine at what ASBR's the BGP route is being learned. Next, the I2RS controller could interrogate those routers BGP policies to determine the root cause of why the route was either not learned or not preferred in BGP. Finally, if necessary, the I2RS controller(s) could amend BGP policies and push them out to BGP routers to permit the BGP route or make it a preferred route according to the BGP path selection algorithm.

### 5.3. BGP Protocol Statistics

There are a variety of statistics related to the operation of BGP that are invaluable to network operators. These statistics generally help operators, and developers, understand the present state and future scalability of BGP.

One statistic that is invaluable to operators is the current number of BGP routes learned through an eBGP session. Operators then apply a command against each eBGP session to limit the maximum number of BGP routes that may be learned through that eBGP session before a warning message is triggered and/or the eBGP session is torn down completely. This configuration capability is often referred to as a "max-prefix limit". This command must be routinely audited and, if necessary, adjusted in order to not trigger a false warning or teardown due to the natural organic growth in BGP routes learned from a given BGP neighbor.

I2RS controllers could provide an invaluable capability to help audit and re-program the "max-prefix limit" on a periodic basis, which is generally once per day. Specifically, the first task would be for an I2RS controller to validate that there is a "max-prefix limit" applied to every eBGP session. (If there is not, that should either trigger a red alarm to the NOC to manually fix this condition or for the I2RS controller to automatically apply a "max-prefix limit" that would alleviate this hazardous condition). Assuming there is a "max-prefix limit" already in place, the I2RS controller would simultaneously retrieve, from each BGP router, the current number of BGP routes learned through a BGP session and value used for the "max-prefix limit" on that same BGP session. These two values could then be handed off to an application that determines if adjustments in the "max-prefix limit" value are required for each BGP session. The application would then notify the I2RS controller of the subset of eBGP sessions and their associated change in "max-prefix limit" value, whereby the I2RS controller would then adjust the BGP protocol configuration on each requisite BGP router in the network. Finally, it should be noted that the above is just one method whereby "max-prefix limit" values are adjusted. It's similarly possible that the BGP routers may, through the I2RS, pull the "max-prefix limit" values for each eBGP neighbor they have onboard on a periodic basis and validate their accuracy.

The above is just one use case related to BGP protocol statistics. There are wealth of other BGP protocol statistics or state information that would be invaluable to have programmatic visibility into that operators do not have today.

## 6. Security Considerations

The BGP use cases described in this document assumes use of I2RS's programmatic interfaces described in the I2RS framework mentioned in [I-D.ward-i2rs-framework]. This document does not change the underlying security issues inherent in the existing [I-D.ward-i2rs-framework].

## 7. Acknowledgements

TBD.

## 8. References

### 8.1. Normative References

- [I-D.ward-i2rs-framework]  
Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Framework", draft-ward-i2rs-framework-00 (work in progress), February 2013.
- [RFC1997] Chandrasekeran, R., Traina, P., and T. Li, "BGP Communities Attribute", RFC 1997, August 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2629] Rose, M.T., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
- [RFC3392] Chandra, R. and J. Scudder, "Capabilities Advertisement with BGP-4", RFC 3392, November 2002.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.
- [RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.
- [RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, February 2006.
- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, January 2007.

## 8.2. Informative References

- [I-D.ietf-grow-ops-reqs-for-bgp-error-handling]  
Shakir, R., "Operational Requirements for Enhanced Error Handling Behaviour in BGP-4", draft-ietf-grow-ops-reqs-for-bgp-error-handling-06 (work in progress), December 2012.
- [I-D.mcpherson-irr-routing-policy-considerations]  
McPherson, D., Amante, S., Osterweil, E., and L. Blunk, "IRR & Routing Policy Configuration Considerations", draft-mcpherson-irr-routing-policy-considerations-01 (work in progress), September 2012.
- [RFC2622] Alaettinoglu, C., Villamizar, C., Gerich, E., Kessens, D., Meyer, D., Bates, T., Karrenberg, D., and M. Terpstra, "Routing Policy Specification Language (RPSL)", RFC 2622, June 1999.
- [RFC2858] Bates, T., Rekhter, Y., Chandra, R., and D. Katz, "Multiprotocol Extensions for BGP-4", RFC 2858, June 2000.
- [RFC5156] Blanchet, M., "Special-Use IPv6 Addresses", RFC 5156, April 2008.
- [RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, August 2009.
- [RFC5735] Cotton, M. and L. Vegoda, "Special Use IPv4 Addresses", BCP 153, RFC 5735, January 2010.

## Authors' Addresses

Keyur Patel  
Cisco Systems  
170 W. Tasman Drive  
San Jose, CA 95134  
USA

Email: keyupate@cisco.com

Rex Fernando  
Cisco Systems  
170 W. Tasman Drive  
San Jose, CA 95134  
USA

Email: [rex@cisco.com](mailto:rex@cisco.com)

Hannes Gredler  
Juniper Networks  
1194 N. Mathilda Ave  
Sunnyvale, CA 94089  
USA

Email: [hannes@juniper.net](mailto:hannes@juniper.net)

Shane Amante  
Level 3 Communications, Inc.  
1025 Eldorado Blvd  
Broomfield, CO 80021  
USA

Email: [shane@level3.net](mailto:shane@level3.net)

Network Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: January 16, 2014

J. Medved  
Cisco  
N. Bahadur  
Juniper Networks  
A. Clemm  
Cisco  
H. Ananthakrishnan  
Juniper Networks  
July 15, 2013

An Information Model for Network Topologies  
draft-medved-i2rs-topology-im-00.txt

Abstract

This document defines the information model for network topologies.

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 January 16, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

## Table of Contents

1. Introduction . . . . .	2
2. Definitions and Acronyms . . . . .	4
3. Network Topology Model Overview . . . . .	4
4. Network Topology Information Model . . . . .	5
4.1. Base Model: the Network-Topology Component . . . . .	6
4.2. Layer 3 Unicast Topology (IGP) Extensions . . . . .	9
4.2.1. The L3-Unicast-Topology Component . . . . .	9
4.2.2. The OSPF-Topology Component . . . . .	11
4.2.3. The IS-IS-Topology Component . . . . .	12
4.2.4. The TED (Traffic Engineering Data) Component . . . . .	13
5. Security Considerations . . . . .	14
6. Contributors . . . . .	14
7. Acknowledgements . . . . .	14
8. References . . . . .	15
8.1. Normative References . . . . .	15
8.2. Informative References . . . . .	15

## 1. Introduction

This document introduces an information model for network topologies. The model allows applications to have a holistic view of an entire network. [I-D.amante-i2rs-topology-use-cases] describes an entity - the Topology Manager - that would create a cohesive, abstracted model of the network and expose it to applications via northbound API.

The information model can be related to a corresponding data model, for example, a data model defined in YANG [RFC6020], defining the actual data that is exchanged across specific interfaces. On the relationship between information and data models, please refer to [RFC3444].

In order to capture information that is specific to different network topology types, this document defines an abstract (basic) topology model that can be extended and adapted. As a result, the information model is generic in nature and can be applied to many network topologies. Applications can operate on any topology at a generic level where specifics of particular topology types are not required, and at a topology-specific level when those specifics come into play. Specific topology types that are covered in this document include Layer 3 Unicast IGP, IS-IS, and OSPF. We also define the information model for traffic engineering (TE) data. Adaptations and extensions to other types of topologies (such as Layer 2 topology or OpenFlow topology) are possible, using similar model patterns to the ones that are illustrated.

This revision of the document focuses on the "live" topology information model ([I-D.amante-i2rs-topology-use-cases], Section 3.2). The "inventory" and "statistics collection" information models will be addressed separately.

The information model contains several components:

**Network-Topology** contains a generic network topology model. It defines a network topology at its most general level of abstraction. It models aspects such as nodes and edges that constitute a topology graph, as well as termination points contained in the nodes that actually terminate edges of the graph. A network can contain multiple topologies, for example topologies at different network layers or overlay topologies. The model therefore allows to show relationships between topologies, as well as dependencies between nodes and termination points across topologies.

**L3-Unicast-IGP-Topology** applies the general network topology model to Layer 3 Unicast IGP topologies. It extends the general topology with information specific to Layer 3 Unicast IGP. In doing so, it also illustrates the extension patterns associated with extending respectively extending the general topology model to meet the needs of a specific topology.



OSPF-Topology Module "ospf-topology" defines a topology model for OSPF, building on and extending the Layer 3 Unicast IGP topology model. It serves as an example of how the general topology model can be refined across multiple levels.

IS-IS-Topology defines a topology model for IS-IS, again building on and extending the Layer 3 Unicast IGP topology model.

TED defines information kept in the Traffic Engineering Database (TED) that is leveraged by IS-IS and OSPF topologies.

## 2. Definitions and Acronyms

Data model: An abstract model of a conceptual domain that is intended for implementors and contains enough specifics to result in interoperable implementations and data representations

Datastore: A conceptual store of instantiated management information, with individual data items represented by data nodes which are arranged in hierarchical manner.

IGP: Interior Gateway Protocol

Information Model: An abstract model of a conceptual domain, independent of a specific implementations or data representation

IS-IS: Intermediate System to Intermediate System protocol

LSP: Label Switched Path

OSPF: Open Shortest Path First, a link state routing protocol

RBNF: Routing Backus-Naur Form

URI: Uniform Resource Identifier

SRLG: Shared Risk Link Group

TED: Traffic Engineering Database

## 3. Network Topology Model Overview

This section provides an overview of the network topology model. We start with the structure of the foundational model that represents a generic topology. Subsequently, an overview of selected specific topologies is given - Layer 3 Unicast IGP, OSPF, and IS-IS, respectively. Throughout the document, selected design choices are explained and the pattern that should be applied to extend the model to new types of topologies is presented.

The network topology model is defined by the following components, whose relationship is roughly depicted in the figure below.

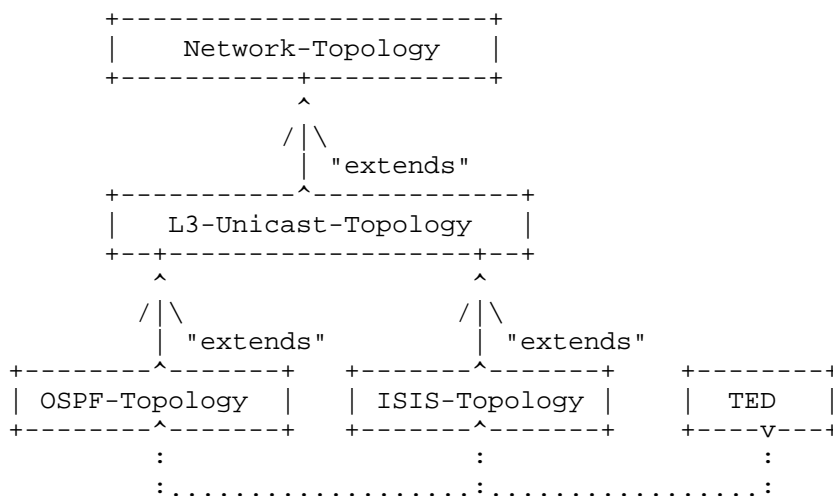


Figure 1: Overall model structure

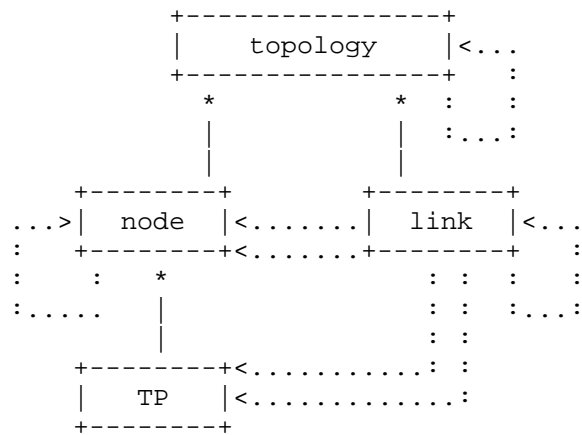
The Network-Topology component defines the basic network topology model. The L3-Unicast-Topology module extends this model with additional definitions needed to represent Layer 3 Unicast IGP topologies. This component in turn is extended by OSPF-Topology and ISIS-Topology components providing additional definitions for OSPF and IS-IS topologies, respectively. The TED component, used by both OSPF-Topology and ISIS-Topology components, contains a set of auxiliary definitions related to traffic engineering.

#### 4. Network Topology Information Model

This section specifies the network topology information model in Routing Backus-Naur Form (RBNF, [RFC5511]). It also provides diagrams of the main entities that the information model is comprised of.

## 4.1. Base Model: the Network-Topology Component

The following diagram contains an informal graphical depiction of the main elements of the information model:



Roughly speaking, the basic information model works as follows: A topology contains nodes and links. Each node in turn contains termination points. A link connects two nodes (a source and a destination), terminating on each at a termination point. Nodes can map onto and be supported by other nodes, while links can map onto and be supported by other links. Topologies can map onto other, underlay topologies.

The information model for the Network-Topology component is more formally shown in the following diagram.

```

<network-topology> ::= (<topology>...)

<topology> ::= <TOPOLOGY_IDENTIFIER>
               (<node>
                (<link>...)
                [<topology-type>]
                [<underlay-topologies>]
                [<topology-extension>])

<topology-type> ::= (<IGP> [<igp-topology-type>]) |
                   (<BGP> [<bgp-topology-type>])
<igp-topology-type> ::= <OSPF> | <ISIS>

<bgp-topology-type> ::= <>
  
```

```

<underlay-topologies> ::= (<TOPOLOGY_IDENTIFIER>...)

<topology-extension> ::= <igp-topology-extension> |
                        <bgp-topology-extension> |
                        ...

<node> ::= <NODE_IDENTIFIER>
          (<termination-point>...)
          [<supporting-nodes>]
          [<node-extension>]

<termination-point> ::= <TERMINATION_POINT_IDENTIFIER>
                       [<supporting-termination-points>]
                       [<igp-termination-point>]

<supporting-termination-points> ::=
    (<TERMINATION_POINT_IDENTIFIER>...)

<supporting-nodes> ::= (<NODE_IDENTIFIER>...)

<node-extension> ::= <igp-node-extension> |
                    <bgp-node-extension> |
                    ...

<link> ::= <LINK_IDENTIFIER>
          <source>
          <destination>
          [<supporting-links>]
          [<link-extension> ]

<source> ::= <termination-point-reference>
<destination> ::= <termination-point-reference>

<termination-point-reference> ::= <NODE_IDENTIFIER>
                                <TERMINATION_POINT_IDENTIFIER>

<supporting-links> ::= (<LINK_IDENTIFIER>...)

<link-extension> ::= <igp-link-extension> |
                    <bgp-link-extension> |
                    ...

```

The elements of the Network-Topology information model are as follows:

- o A network can contain multiple topologies. Each topology is captured in its own list element, distinguished via a topology-id.

- o A topology has a certain type, such as OSPF or IS-IS. A topology can even have multiple types simultaneously. The type, or types, are captured in the list of "topology-type" components.
- o A topology can in turn be part of a hierarchy of topologies, building on top of other topologies. Any such topologies are captured in list "underlay-topology".
- o Furthermore, a topology contains nodes and links, each captured in their own list.
- o A node has a node-id. This distinguishes the node from other nodes in the list. In addition, a node has a list of termination points, used to terminate links. An examples of a termination point might be a physical or logical port or, more generally, an interface. Also, a node can in turn map onto other nodes in an underlay topology. This is captured in list "supporting-node".
- o A link is identified by a link-id, uniquely identifying the link within the topology. Links are point-to-point and unidirectional. Accordingly, a link contains a source and a destination. Both source and destination reference a corresponding node, as well as a termination point on that node. Analogous to a node, a link can in turn map onto other links an underlay topology. This is captured in list "supporting-link".
- o The topology, node and link elements can be extended with topology-specific components (topology-extensions, node-extension and link-extension, respectively). This document defines extensions for the L3 unicast topology.

The topology model includes links that are point-to-point and unidirectional. It does not directly support multipoint and bidirectional links. While this may appear as a limitation, it does keep the model simple, generic, and allows it to very easily be subjected applications that make use of graph algorithms. Bi-directional connections can be represented through pairs of unidirectional links. By introducing hierarchies of nodes, with nodes at one level mapping onto a set of other nodes at another level, and the introducing new links for nodes at that level, topologies with connections representing non-point-to-point communication patterns can be represented.

To minimize assumptions of what a topology might actually represent, mappings between topologies, nodes, links, and termination points are kept strictly generic. For example, no assumptions are made whether a termination point actually refers to an interface, or whether a node refers to a specific "system" or device; the model at this

generic level makes no provisions for that. Any greater specifics about mappings between upper and lower layers can be captured in extending modules.

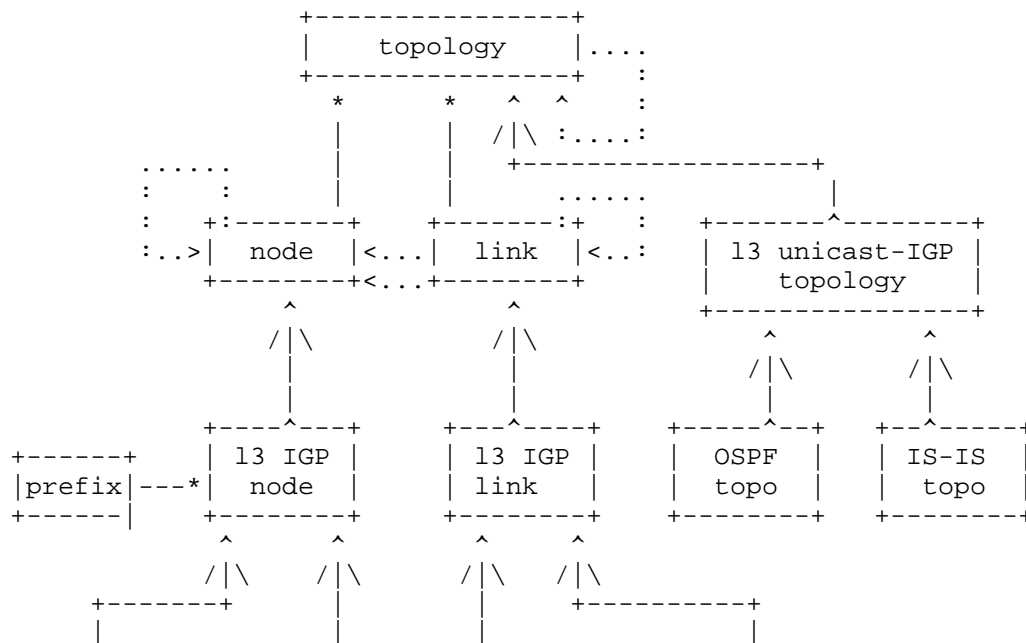
Links are terminated by a single termination point, not sets of termination points. Connections involving multihoming or link aggregation schemes need to be modeled using multiple point-to-point links, then define a link at a higher layer that is supported by those individual links.

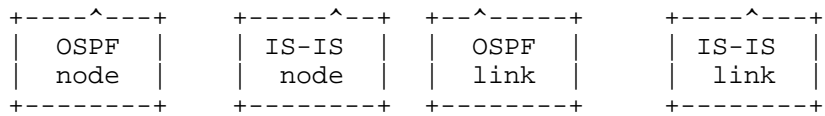
In a hierarchy of topologies, there are nodes mapping to nodes, links mapping to links, and termination points mapping to termination points. Some of this information is redundant. Specifically, with the link-to-links mapping known, and the termination points of each link known, maintaining separate termination point mapping information is not needed but can be derived via transitive closure.

## 4.2. Layer 3 Unicast Topology (IGP) Extensions

### 4.2.1. The L3-Unicast-Topology Component

In order to represent a general Layer 3 Unicast IGP topology, the basic network topology model needs to be extended. The corresponding extensions are introduced in a component, whose structure is informally depicted in the following diagram.





Roughly speaking, layer 3 IGP topology refines the generic topology, and layer 3 IGP nodes, IGP links, and IGP termination points (not depicted) refine the generic nodes, links, and termination points. In addition, layer 3 IGP nodes can contain prefixes. The pattern recurses with OSPF and IS-IS topologies, which are in turn derived from the corresponding layer 3 IGP entities.

A more formal depiction in RBNF format follows below:

```

<igp-topology> ::= <TOPOLOGY_NAME>
                  [(<FLAGS>...)]
                  [(<ospf-topology> | <isis-topology>)]

<igp-node> ::= <NODE_NAME>
               (<router-id>...)
               [(<prefix>...)]
               [(<FLAGS>...)]
               [(<isis-node> | <ospf-node>)]

<router-id> ::= <ip-address> | <NUMBER>

<ip-address> ::= (<IPV4><IPV4_ADDRESS>) | (<IPV6><IPV6_ADDRESS>)

<prefix> ::= <ip-route>
            <METRIC>
            (<FLAGS>...)
            [(<ospf-prefix> | <isis-prefix>)]

<ip-route> ::= <ip-address>
              <PREFIX_LENGTH>

<igp-link> ::= <LINK_NAME>
               <METRIC>
               [(<FLAGS>...)]
               [ (<isis-link> | <ospf-link>) ]

<igp-termination-point> ::= (<ip-address>...) | <NUMBER>

```

The model extends the original network-topology model as follows:

- o A new topology type is introduced, igp-topology.

- o Additional topology attributes are introduced. This allows to introduce additional flags in extending modules that are associated with specific IGP topologies, without needing to revise this component.
- o Additional data objects for nodes are introduced by extending the "node" list of the network topology module. New objects include again a set of flags, as well as a list of prefixes. Each prefix in turn includes an ip prefix, a metric, and a prefix-specific set of flags.
- o Links are extended as well with a set of parameters, allowing to associate a link with an IGP name, another set of flags, and a link metric.

#### 4.2.2. The OSPF-Topology Component

OSPF is the next type of topology represented in the model. OSPF represents a particular type of Layer 3 Unicast IGP. Accordingly, the Layer 3 Unicast IGP topology model needs to be extended. The corresponding extensions are introduced in a separate component "ospf-topology", whose structure is depicted in the following diagram. For the most part, this module extends the "l3-unicast-igp-topology" component.

```

<ospf-topology> ::= <AREA_IDENTIFIER>

<ospf-node> ::= <ospf-router-type>
                [<DR_INTERFACE_IDENTIFIER>]
                [( <MULTI_TOPOLOGY_IDENTIFIER>... )]
                [<ospf-node-capabilities>]
                [<ted-node>]

<ospf-router-type> ::= <ABR> | <ASBR> | <INTERNAL> | <PSEUDONODE>

<ospf-node-capabilities> ::= <>

<ospf-link> ::= [<MULTI_TOPOLOGY_IDENTIFIER>]
                [<ted-link>]

<ospf-prefix> ::= [<forwarding-address>]

<forwarding-address> ::= <IPV4><IPV4_ADDRESS>

```

The module extends the l3-unicast-igp-topology as follows:

- o A new topology type is introduced, ospf-topology-type.



- o Additional topology attributes are introduced, which extend the igp-topology-attributes of the l3-unicast-igp-topology component. The attributes include an OSPF area-id identifying the area.
- o Additional data objects for nodes are introduced by extending the igp-node-attributes of the l3-unicast-igp-topology component. New objects include router-type, de-interface-id for pseudonodes, list of multi-topology-ids, OSPF node capabilities and traffic engineering attributes.
- o Links are extended with multi-topology-id and traffic engineering link attributes.
- o Prefixes are extended with OSPF specific forwarding address.

#### 4.2.3. The IS-IS-Topology Component

IS-IS is another type of Layer 3 Unicast IGP. Like OSPF topology, IS-IS topology is defined in a separate component, "isis-topology", which extends "l3-unicast-igp-topology".

```
<isis-topology> ::= <NET_IDENTIFIER>

<isis-node> ::= <isis-router-type> <iso>
               (<NET_IDENTIFIER>...)
               [( <MULTI_TOPOLOGY_IDENTIFIER>...)]
               [<ted-node>]

<iso> ::= <ISO_SYSTEM_ID> <ISO_PSEUDONODE_ID>
<isis-router-type> ::= <LEVEL_2> | <LEVEL_1> | <LEVEL_2_1>

<isis-link> ::= [<MULTI_TOPOLOGY_IDENTIFIER>]
               [<ted-link>]

<isis-prefix> ::= <>
```

The module extends the l3-unicast-igp-topology component as follows:

- o A new topology type is introduced, "isis-topology-type".
- o Additional topology attributes are introduced. The attributes include an ISIS NET-id identifying the area.

- o Additional data objects for nodes are introduced by extending "igp-node-attributes" of the l3-unicast-igp-topology component. New objects include router-type, iso-system-id to identify the router, list of multi-topology-id, list of NET ids and traffic engineering attributes.
- o Links are extended with multi-topology-id and traffic engineering link attributes.

In addition, the module extends IGP node and link with ISIS attributes.

#### 4.2.4. The TED (Traffic Engineering Data) Component

Traffic Engineering Data is required both by OSPF and IS-IS, which are defined in separate components. Information shared by both is defined in another component, "TED". This component defines a set of groupings with auxiliary information required and shared by those other components.

```

<ted-node> ::= <te-router-id>
               <local-address>
               <pcc-capabilities>

<te-router-id> ::= [<IPV4><IPV4_ADDRESS>]
                  [<IPV6><IPV6_ADDRESS>]

<local-address> ::= [((<IPV4><IPV4_ADDRESS>)...)] |
                  [((<IPV6><IPV6_ADDRESS><PREFIX_OPTION>)...)]

<pcc-capabilities> ::= <>

<ted-link> ::= <COLOR>
               <MAX_LINK_BANDWIDTH>
               <MAX_RESV_LINK_BANDWIDTH>
               (<UNRESERVED_BANDWIDTH>...)
               <TE_DEFAULT_METRIC>
               [<srlg-attributes>]

<srlg-attributes> ::= ( <interface-switching-capabilities>...)
                    (<SRLG_VALUE>...)
                    <LINK_PROTECTION_TYPE>

<interface-switching-capabilities> ::= <switching-capabilities>
                                       <ENCODING>
                                       (<MAX_LSP_BANDWIDTH>...)
                                       [<packet-switch-capable>]
                                       [<tdm-capable>]

```

```
<packet-switch-capable> ::= <MINIMUM_LSP_BANDWIDTH>  
                             <INTERFACE_MTU>  
  
<time-division-switch-capable> ::= <MINIMUM_LSP_BANDWIDTH>  
                                    <INDICATION>  
  
<switching-capabilities> ::= <>
```

This module details traffic-engineering node and link attributes:

- o TED node attributes include te-router-id for IPv4 and IPv6, local IPv4 and IPv6 addresses and path computation client capabilities. The path computation client capabilities in turn include a bit vector for various path computation capabilities.
- o TED link attributes comprise link color, max-link-bandwidth, max-resv-link-bandwidth, unreserved bandwidth and re-metric. They also include SRLG attributes which contains interface switching capabilities, a list of SRLG values and link protection type. The interface switching capabilities in turn contains switching capability, encoding, max-lsp-bandwidth and interface switching specific attributes.

## 5. Security Considerations

TBD

## 6. Contributors

The model presented in this paper was contributed to by more people than can be listed on the author list. Additional contributors include:

- o Ken Gray, Juniper Networks
- o Tom Nadeau, Juniper Networks
- o Aleksandr Zhdankin, Cisco
- o Tony Tkacik, Cisco
- o Robert Varga, Pantheon Technologies

## 7. Acknowledgements

We wish to acknowledge the helpful contributions, comments, and suggestions that were received from many people. We'd also like to

thank the people that contributed to the topology information model during the I2RS Interim meeting in April 2013 - Shane Amante, Andy Bierman, Ed Crabbe, Adrian Farrel, and Joel Halpern.

## 8. References

### 8.1. Normative References

- [RFC5511] Farrel, A., "Routing Backus-Naur Form (RBNF): A Syntax Used to Form Encoding Rules in Various Routing Protocol Specifications", RFC 5511, April 2009.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

### 8.2. Informative References

- [I-D.amante-i2rs-topology-use-cases] Amante, S., Medved, J., Previdi, S., and T. Nadeau, "Topology API Use Cases", draft-amante-i2rs-topology-use-cases-00 (work in progress), February 2013.
- [RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, January 2003.

## Authors' Addresses

Jan Medved  
Cisco

EMail: jmedved@cisco.com

Nitin Bahadur  
Juniper Networks

EMail: nitinb@juniper.net>

Alexander Clemm  
Cisco

EMail: alex@cisco.com

Hariharan Ananthakrishnan  
Juniper Networks

EMail: hanantha@juniper.net

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

N. Bahadur, Ed.  
R. Folkes, Ed.  
Juniper Networks, Inc.  
S. Kini  
Ericsson  
J. Medved  
Cisco  
July 15, 2013

Routing Information Base Info Model  
draft-nitinb-i2rs-rib-info-model-01

Abstract

Routing and routing functions in enterprise and carrier networks are typically performed by network devices (routers and switches) using a routing information base (RIB). Protocols and configuration push data into the RIB and the RIB manager install state into the hardware; for packet forwarding. This draft specifies an information model for the RIB to enable defining a standardized data model. Such a data model can be used to define an interface to the RIB from an entity that may even be external to the network device. This interface can be used to support new use-cases being defined by the IETF I2RS WG.

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 January 16, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	4
1.1. Conventions used in this document . . . . .	6
2. RIB data . . . . .	6
2.1. RIB definition . . . . .	6
2.2. Routing tables . . . . .	7
2.3. Route . . . . .	8
2.4. Nexthop . . . . .	9
2.4.1. Nexthop types . . . . .	10
2.4.2. Nexthop list attributes . . . . .	11
2.4.3. Nexthop content . . . . .	11
2.4.4. Nexthop attributes . . . . .	12
2.4.5. Nexthop vendor attributes . . . . .	13
2.4.6. Special nexthops . . . . .	13
3. Reading from the RIB . . . . .	14
4. Writing to the RIB . . . . .	14
5. Events and Notifications . . . . .	14
6. RIB grammar . . . . .	15
7. Using the RIB grammar . . . . .	17
7.1. Using route preference and metric . . . . .	18
7.2. Using different nexthops types . . . . .	18
7.2.1. Tunnel nexthops . . . . .	18
7.2.2. Replication lists . . . . .	18
7.2.3. Weighted lists . . . . .	19
7.2.4. Protection lists . . . . .	19
7.2.5. Nexthop chains . . . . .	20
7.2.6. Lists of lists . . . . .	20
7.3. Performing multicast . . . . .	20
7.4. Solving optimized exit control . . . . .	21
8. RIB operations at scale . . . . .	21
8.1. RIB reads . . . . .	22
8.2. RIB writes . . . . .	22
8.3. RIB events and notifications . . . . .	22
9. Security Considerations . . . . .	22
10. IANA Considerations . . . . .	22
11. Acknowledgements . . . . .	22
12. References . . . . .	23
12.1. Normative References . . . . .	23
12.2. Informative References . . . . .	23
Authors' Addresses . . . . .	24



## 1. Introduction

Routing and routing functions in enterprise and carrier networks are traditionally performed in network devices. Traditionally routers run routing protocols and the routing protocols (along with static config) populates the Routing information base (RIB) of the router. The RIB is managed by the RIB manager and it provides a north-bound interface to its clients i.e. the routing protocols to insert routes into the RIB. The RIB manager consults the RIB and decides how to program the forwarding information base (FIB) of the hardware by interfacing with the FIB-manager. The relationship between these entities is shown in Figure 1.

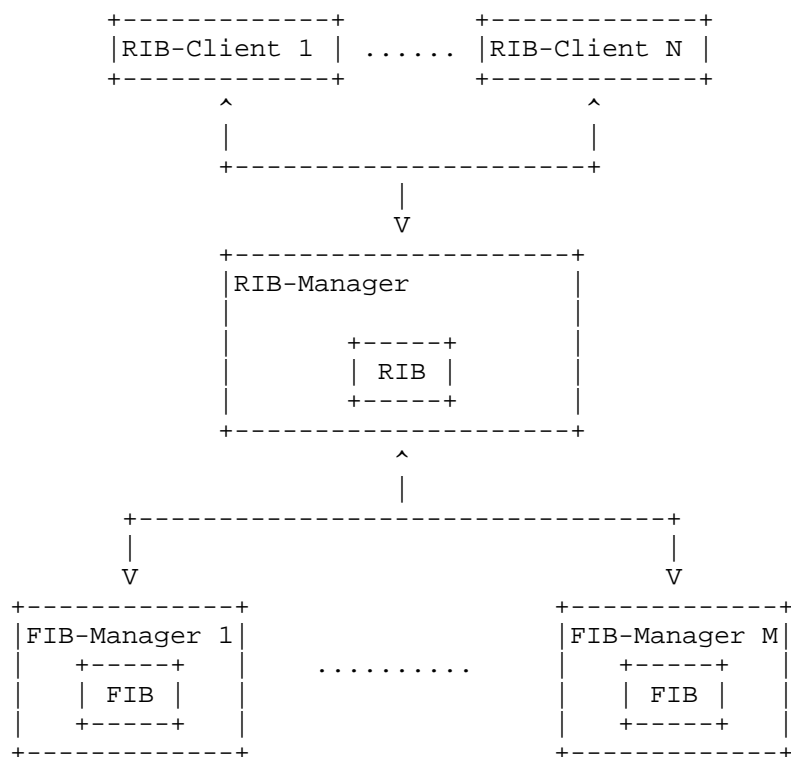


Figure 1: RIB-Manager, RIB-Clients and FIB-Managers

Routing protocols are inherently distributed in nature and each router makes an independent decision based on the routing data received from its peers. With the advent of newer deployment paradigms and the need for specialized applications, there is an emerging need to guide the router's routing function [I-D.atlas-i2rs-problem-statement]. Traditional network-device

protocol-based RIB population suffices for most use cases where distributed network control works. However there are use cases in which the network admins today configure static routes, policies and RIB import/export rules on the routers. There is also a growing list of use cases [I-D.white-i2rs-use-case], [I-D.hares-i2rs-use-case-vn-vc] in which a network admin might want to program the RIB based on data unrelated to just routing (within that network's domain). It could be based on routing data in adjacent domain or it could be based on load on storage and compute in the given domain. Or it could simply be a programmatic way of creating on-demand dynamic overlays between compute hosts (without requiring the hosts to run traditional routing protocols). If there was a standardized programmatic interface to a RIB, it would fuel further networking applications targeted towards specific niches.

A programmatic interface to the RIB involves 2 types of operations - reading what's in the RIB and adding/modifying/deleting contents of the RIB. [I-D.white-i2rs-use-case] lists various use-cases which require read and/or write manipulation of the RIB.

In order to understand what is in a router's RIB, methods like per-protocol SNMP MIBs and show output screen scraping are being used. These methods are not scalable, since they are client pull mechanisms and not proactive push (from the router) mechanisms. Screen scraping is error prone (since output can change) and vendor dependent. Building a RIB from per-protocol MIBs is error prone since the MIB data represents protocol data and not the exact information that went into the RIB. Thus, just getting read-only RIB information from a router is a hard task.

Adding content to the RIB from an external entity can be done today using static configuration support provided by router vendors. However the mix of what can be modified in the RIB varies from vendor to vendor and the way of configuring it is also vendor dependent. This makes it hard for an external entity to program a multi-vendor network in a consistent and vendor independent way.

The purpose of this draft is to specify an information model for the RIB. Using the information model, one can build a detailed data model for the RIB. And that data model could then be used by an external entity to program a router.

The rest of this document is organized as follows. Section 2.1 goes into the details of what constitutes and can be programmed in a RIB. Section 5 provides a high-level view of the events and notifications going from a network device to an external entity, to update the external entity on asynchronous events. The RIB grammar is specified in Section 6. Examples of using the RIB grammar are shown in

## Section 7.

### 1.1. Conventions used in this document

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].

### 2. RIB data

This section describes the details of a RIB. It makes forward references to objects in the RIB grammar (Section 6).

#### 2.1. RIB definition

A RIB is a logical construct controlled by an external entity. A RIB contains one or more routing instances. On a network device, a RIB is uniquely identified by its name. A routing instance can be in only 1 RIB. A routing instance, in the context of the RIB information model, is a collection of routing tables, interfaces, and routing parameters. A routing instance creates a logical slice of the router and allows different logical slices; across a set of routers; to communicate with other each. Layer 3 Virtual Private Networks (VPN), Layer 2 VPNs (L2VPN) and Virtual Private Lan Service (VPLS) can be modeled as routing instances. Note that modeling a Layer 2 VPN using a routing instance only models the Layer-3 (RIB) aspect and does not model any layer-2 information (like ARP) that might be associated with the L2VPN.

The set of interfaces indicates which interfaces this routing instance has control over. The routing tables specify how incoming traffic is to be forwarded. And the routing parameters control the information in the routing tables. The intersection set of interfaces of 2 routing instances MUST be the null set. In other words, an interface should not be present in 2 routing instances. Thus a routing instance describes the routing information and parameters across a set of interfaces.

A routing instance MUST contain the following mandatory fields.

- o INSTANCE\_NAME: A routing instance is identified by its name, INSTANCE\_NAME.
- o INSTANCE\_DISTINGUISHER: Each routing instance must have a unique distinguisher associated with it. It enables one to distinguish routes across routing instances. The route distinguisher SHOULD be unique across all routing instances in a given network device. How the INSTANCE\_DISTINGUISHER is allocated and kept unique is outside the scope of this document. The instance distinguisher

maps well to BGP route-distinguisher for virtual private networks (VPNs). However, the same concept can be used for other use-cases as well.

- o routing-table-list: This is the list of routing tables associated with this routing instance. Each routing instance can have multiple tables to represent routes of different types. For example, one would put IPv4 routes in one table and MPLS routes in another table.
- A routing instance MAY contain the following optional fields.
- o interface-list: This represents the list of interfaces in this routing instance. The interface list helps constrain the boundaries of packet forwarding. Packets coming on these interfaces are directly associated with the given routing instance. The interface list contains a list of identifiers, with each identifier uniquely identifying an interface.
  - o ROUTER\_ID: The router-id field identifies the router. This field is to be used if one wants to virtualize a physical router into multiple virtual routers. Each virtual router will have a unique router-id.
  - o ISO\_SYSTEM\_ID: For IS-IS to operate on a router, a system identifier is needed. This represents the same.
  - o as-data: The as-data fields is used when the routes in this instance are to be tagged with certain autonomous system (AS) characteristics. The RIB manager can use AS length as one of the parameters for making path selection. as-data consists of a AS number and an optional Confederation AS number ([RFC5065]).

## 2.2. Routing tables

A routing table is an entity that contains routes. A routing table is identified by its name. The name MUST be unique within a RIB. All routes in a given routing table MUST be of the same type (e.g. IPv4). Each routing table MUST belong to some routing instance.

A routing table can be tagged with a MULTI\_TOPOLOGY\_ID. If a routing instance is divided into multiple logical topologies, then the multi-topology field is used to distinguish one topology from the other, so as to keep routes from one topology independent of routes from another topology.

If a routing instance contains multiple tables of the same type (e.g. IPv4), then a MULTI\_TOPOLOGY\_ID MUST be associated with each such table. Multiple tables are useful when describing multiple topology IGP (Interior Gateway Protocol) networks (see [RFC4915] and [RFC5120]). In a given routing instance, MULTI\_TOPOLOGY\_ID MUST be unique across routing tables of the same type.

Each route table can be optionally associated with a

ENABLE\_IP\_RPF\_CHECK attribute that enables Reverse path forwarding (RPF) checks on all IP routes in that table. Reverse path forwarding (RPF) check is used to prevent spoofing and limit malicious traffic. For IP packets, the IP source address is looked up and the rpf interface(s) associated with the route for that IP source address is found. If the incoming IP packet's interface matches one of the rpf interface(s), then the IP packet is forwarded based on its IP destination address; otherwise, the IP packet is discarded.

### 2.3. Route

A route is essentially a match condition and an action following the match. The match condition specifies the kind of route (IPv4, MPLS, etc.) and the set of fields to match on. This document specifies the following match types:

- o IPv4: Match on destination IP in IPv4 header
- o IPv6: Match on destination IP in IPv6 header
- o MPLS: Match on a MPLS tag
- o MAC: Match on ethernet destination addresses
- o Interface: Match on incoming interface of packet
- o IP multicast: Match on (S, G) or (\*, G), where S and G are IP prefixes

Each route can have associated with it one or more optional route attributes.

- o ROUTE\_PREFERENCE: This is a numerical value that allows for comparing routes from different protocols. It is also known as administrative-distance. The lower the value, the higher the preference. For example there can be an OSPF route for 192.0.2.1/32 with a preference of 5. If a controller programs a route for 192.0.2.1/32 with a preference of 2, then the controller entered route will be preferred by the RIB manager. Preference should be used to dictate behavior. For more examples of preference, see Section 7.1.
- o ROUTE\_METRIC: Route preference is used for comparing routes from different protocols. Route metric is used for comparing routes learned by the same protocol. If a controller wishes to program 2 or more routes to the same destination, then it can use the metric field to disambiguate the 2 routes. For more examples, see Section 7.1.
- o LOCAL\_ONLY: This is a boolean value. If this is present, then it means that this route should not be exported into other RIBs or other route tables.
- o rpf-check-interface: Reverse path forwarding (RPF) check is used to prevent spoofing and limit malicious traffic. For IP packets, the IP source address is looked up and the rpf-check-interface associated with the route for that IP source address is found. If the incoming IP packet's interface matches one of the rpf-check-

interfaces, then the IP packet is forwarded based on its IP destination address; otherwise, the IP packet is discarded. For MPLS routes, there is no source address to be looked up, so the usage is slightly different. For an MPLS route, a packet with the specified MPLS label will only be forwarded if it is received on one of the interfaces specified by the rpf-check-interface. If no rpf-check-interface is specified, then matching packets are no subject to this check. This field overrides the ENABLE\_IP\_RPF\_CHECK flag on the routing table and interfaces provided in this list are used for doing the RPF check.

- o as-path: A route can have an as-path associated with it to indicate which set of autonomous systems has to be traversed to reach the final destination. The as-path attribute can be used by the RIB manager in multiple ways. The RIB manager can choose paths with lower as-path length. Or the RIB manager can choose to not install paths going via a particular AS. How exactly the RIB manager uses the as-path is outside the scope of this document. For details of how the as-path is formed, see Section 5.1.2 of [RFC4271] and Section 3 of [RFC5065].
- o route-vendor-attributes: Vendors can specify vendor-specific attributes using this. The details of this field is outside the scope of this document.

#### 2.4. Nexthop

A nexthop represents an object or action resulting from a route lookup. For example, if a route lookup results in sending the packet out a given interface, then the nexthop represents that interface.

Nexthops can be fully resolved nexthops or unresolved nexthop. A resolved nexthop is something that is ready for installation in the FIB. For example, a nexthop that points to an interface. An unresolved nexthop is something that requires the RIB manager to figure out the final resolved nexthop. For example, a nexthop could point to an IP address. The RIB manager has to resolve how to reach that IP address - is the IP address reachable by regular IP forwarding or by a MPLS tunnel or by both. If the RIB manager cannot resolve the nexthop, then the nexthop stays in unresolved state and is NOT a candidate for installation in the FIB. Future RIB events can cause a nexthop to get resolved (like that IP address being advertised by an IGP neighbor).

The RIB information model allows an external entity to program nexthops that may be unresolved initially. Whenever a unresolved nexthop gets resolved, the RIB manager will send a notification of the same (see Section 5 ).

Nexthops can be identified by an identifier to create a level of

indirection. The identifier is set by the RIB manager and returned to the external entity on request. The RIB data-model SHOULD support a way to optionally receive a nexthop identifier for a given nexthop. For example, one can create a nexthop that points to a BGP peer. The returned nexthop identifier can then be used for programming routes to point to the same nexthop. Given that the RIB manager has created an indirection for that BGP peer using the nexthop identifier, if the transport path to the BGP peer changes, that change in path will be seamless to the external entity and all routes that point to that BGP peer will automatically start going over the new transport path. Nexthop indirection using identifier could be applied to not just unicast nexthops, but even to nexthops that contain chains and nested nexthops (Section 2.4.1).

#### 2.4.1. Nexthop types

This document specifies a very generic, extensible and recursive grammar for nexthops. Nexthops can be

- o Unicast nexthops - pointing to an interface
- o Tunnel nexthops - pointing to a tunnel
- o Replication lists - list of nexthops to which to replicate a packet to
- o Weighted lists - for load-balancing
- o Protection lists - for primary/backup paths
- o Nexthop chains - for chaining headers, e.g. MPLS label over a GRE header
- o Lists of lists - recursive application of the above
- o Indirect nexthops - pointing to a nexthop identifier
- o Special nexthops - for performing specific well-defined functions

It is expected that all network devices will have a limit on recursion and not all hardware will be able to support all kinds of nexthops. RIB capability negotiation becomes very important for this reason and a RIB data-model MUST specify a way for an external entity to learn about the network device's capabilities. Examples of when and how to use various kinds of nexthops are shown in Section 7.2.

Tunnel nexthops allow an external entity to program static tunnel headers. There can be cases where the remote tunnel end-point does not support dynamic signaling (e.g. no LDP support on a host) and in those cases the external entity might want to program the tunnel header on both ends of the tunnel. The tunnel nexthop is kept generic with specifications provided for some commonly used tunnels. It is expected that the data-model will model these tunnel types with complete accuracy.

Nexthop chains can be used to specify multiple headers over a packet, before a packet is forwarded. One simple example is that of MPLS over GRE, wherein the packet has a inner MPLS header followed by a

GRE header followed by an IP header. The outermost IP header is decided by the network device whereas the MPLS header and GRE header are specified by the controller. Not every network device will be able to support all kinds of nexthop chains and an arbitrary number of header chained together. The RIB data-model SHOULD provide a way to expose nexthop chaining capability supported by a given network device.

#### 2.4.2. Nexthop list attributes

For nexthops that are of the form of a list(s), attributes can be associated with each member of the list to indicate the role of an individual member of the list. Two kinds of attributes are specified:

- o PROTECTION\_PREFERENCE: This provides a primary/backup like preference. The preference is an integer value that should be set to 1 or 2. Nexthop members with a preference of 1 are preferred over those with preference of 2. The network device SHOULD create a list of nexthops with preference 1 (primary) and another list of nexthops with preference 2 (backup) and SHOULD pre-program the forwarding plane with both the lists. In case if all the primary nexthops fail, then traffic MUST be switched over to members of the backup nexthop list. All members in a list MUST either have a protection preference specified or all members in a list MUST NOT have a protection preference specified.
- o LOAD\_BALANCE\_WEIGHT: This is used for load-balancing. Each list member MUST be assigned a weight. The weight is a percentage number from 1 to 99. The weight determines how much traffic is sent over a given list member. If one of the members nexthops in the list is not active, then the weight value of that nexthop SHOULD be distributed among the other active members. How the distribution is done is up to the network device and not in the scope of the document. In other words, traffic should always be load-balanced even if there is a failure. After a failure, the external entity SHOULD re-program the nexthop list with updated weights so as to get a deterministic behavior among the remaining list members. To perform equal load-balancing, one MAY specify a weight of "0" for all the member nexthops. The value "0" is reserved for equal load-balancing and if applied, MUST be applied to all member nexthops.

#### 2.4.3. Nexthop content

At the lowest level, a nexthop can point to a:

- o identifier: This is an identifier returned by the network device representing another nexthop or another nexthop chain.



- o EGRESS\_INTERFACE: This represents a physical, logical or virtual interface on the network device.
- o address: This can be an IP address or MAC address or ISO address.
  - \* An optional table name can also be specified to indicate the table in which the address is to be looked up further. One can use the table name field to direct the packet from one domain into another domain. For example, a MPLS packet coming in on an interface would be looked up in a MPLS routing table and the nexthop for that could indicate that we strip the MPLS label and do a subsequent IPv4 lookup in an IPv4 table. By default the table will be the same in which the route lookup was performed.
  - \* An optional egress interface can be specified to indicate which interface to send the packet out on. The egress interface is useful when the network device contains Ethernet interfaces and one needs to perform an ARP lookup for the IP packet.
- o tunnel encap: This can be an encap representing an IP tunnel or MPLS tunnel or others as defined in this document. An optional egress interface can be specified to indicate which interface to send the packet out on. The egress interface is useful when the network device contains Ethernet interfaces and one needs to perform an ARP lookup for the IP packet.
- o logical tunnel: This can be a MPLS LSP or a GRE tunnel (or others as defined in this document), that is represented by a unique identifier (E.g. name).
- o ROUTING\_TABLE\_NAME: This is a routing table that exists in the RIB. A nexthop pointing to a table indicates that the route lookup needs to continue in the specified table. This is a way to perform chained lookups.

#### 2.4.4. Nexthop attributes

Certain information is encoded implicitly in the nexthop and does not need to be specified by the controller. For example, when a IP packet is forwarded out, the IP TTL is decremented by default. Same applies for an MPLS packet. Similarly, when an IP packet is sent over an ethernet interface, any ARP processing is handled implicitly by the network device and does not need to be programmed by an external device.

A nexthop can have some attributes associated with it. The purpose of the attributes is to either override implicit behavior (like that related to TTL processing) or to guide the network device to perform something specific. Vendor specific attributes can also be specified. The details of vendor specific attributes is outside the scope of this document.

#### 2.4.4.1. Nexthop flags

Nexthop flags in a nexthop is an optional attribute that is used to denote specific connotation to hardware. Two common types of operations are specified using nexthop flags.

- o NO\_DECREMENT\_TTL: This indicates that the IPv4 time-to-live field in an IPv4 packet MUST NOT be decremented before the packet is forwarded. This may be applied one when an IPv4 packet is encapsulated in a tunnel (E.g. MPLS) and one wants to hide the fact that the packet is going through a tunnel.
- o NO\_PROPAGATE\_TTL: This indicates that the IPv4 time-to-live field in an IPv4 packet MUST NOT be propagated into an equivalent field, when the IPv4 packet is tunneled. For example, if the IPv4 packet is tunneled over MPLS, then the network device should use the default time-to-live value for the outer MPLS header. This field can also be used to indicate that when a tunnel terminates, one does not propagate the outer header's time-to-live value into the inner header. So, on MPLS tunnel termination, one does not propagate the MPLS TTL value into the IPv4 header.

The TTL nexthop flags can be used to simulate a Pipe model for tunnels. See [RFC3443] for a detailed understanding of Pipe model and Uniform model.

#### 2.4.4.5. Nexthop vendor attributes

This field has been defined for vendor specific extensions. The contents of this field are beyond the scope of this document.

#### 2.4.4.6. Special nexthops

This document specifies certain special nexthops. The purpose of each of them is explained below:

- o DISCARD: This indicates that the network device should drop the packet and increment a drop counter.
- o DISCARD\_WITH\_ERROR: This indicates that the network device should drop the packet, increment a drop counter and send back an appropriate error message (like ICMP error).
- o RECEIVE: This indicates that that the traffic is destined for the network device. For example, protocol packets or OAM packets. All locally destined traffic SHOULD be throttled to avoid a denial of service attack on the router's control plane. An optional rate-limiter can be specified to indicate how to throttle traffic destined for the control plane. The description of the rate-limiter is outside the scope of this document.

### 3. Reading from the RIB

A RIB data-model MUST allow an external entity to read entries, for RIBs created by that entity. The network device administrator MAY allow reading of other RIBs by an external entity through access lists on the network device. The details of access lists are outside the scope of this document.

The data-model MUST support a full read of the RIB and subsequent incremental reads of changes to the RIB. An external agent SHOULD be able to request a full read at any time in the lifecycle of the connection. When sending data to an external entity, the RIB manager SHOULD try to send all dependencies of an object prior to sending that object.

### 4. Writing to the RIB

A RIB data-model MUST allow an external entity to write entries, for RIBs created by that entity. The network device administrator MAY allow writes to other RIBs by an external entity through access lists on the network device. The details of access lists are outside the scope of this document.

When writing an object to a RIB, the external entity SHOULD try to write all dependencies of the object prior to sending that object. The data-model MUST support requesting identifiers for nexthops and collecting the identifiers back in the response.

Route programming in the RIB SHOULD result in a return code that contains the following attributes:

- o Installed - Yes/No (Indicates whether the route got installed in the FIB)
- o Active - Yes/No (Indicates whether a route is fully resolved and is a candidate for selection)
- o Reason - E.g. Not authorized

The data-model MUST specify which objects are modify-able objects. A modify-able object is one whose contents can be changed without having to change objects that depend on it and without affecting any data forwarding. To change a non-modifiable object, one will need to create a new object and delete the old one. For example, routes that use a nexthop that is identifier by a nexthop-identifier should be unaffected when the contents of that nexthop changes.

### 5. Events and Notifications

Asynchronous notifications are sent by the network device's RIB

manager to an external entity when some event occurs on the network device. A RIB data-model MUST support sending asynchronous notifications. A brief list of suggested notifications is as below:

- o Route change notification, with return code as specified in Section 4
- o Nexthop resolution status (resolved/unresolved) notification

## 6. RIB grammar

This section specifies the RIB information model in Routing Backus-Naur Form [RFC5511].

```

<rib> ::= <RIB_NAME> <routing-instance> [<routing-instance> ...]
<routing-instance> ::= <INSTANCE_NAME> <INSTANCE_DISTINGUISHER>
                        [<interface-list>] <routing-table-list>
                        [<ROUTER_ID>] [<ISO_SYSTEM_ID>]
                        [<as-data>]

<as-data> ::= <AS_NUMBER> [<CONFEDERATION_AS>]

<interface-list> ::= (<INTERFACE_IDENTIFIER> ...)

<routing-table-list> ::= (<routing-table> ...)
<routing-table> ::= <ROUTING_TABLE_NAME> <table-family>
                   [<route> ... ] [<MULTI_TOPOLOGY_ID>]
                   [<ENABLE_IP_RPF_CHECK>]
<table-family> ::= <IPV4_TABLE_FAMILY> | <IPV6_TABLE_FAMILY> |
                   <MPLS_TABLE_FAMILY> | <IEEE_MAC_TABLE_FAMILY>

<route> ::= <match> <nexthop-list>
            [<route-attributes>]
            [<route-vendor-attributes>]

<match> ::= <ipv4-route> | <ipv6-route> | <mpls-route> |
            <mac-route> | <interface-route>
<ipv4-route> ::= <IPV4_ADDRESS> <IPV4_PREFIX_LENGTH>
                [<multicast-source-ipv4-address>]
<ipv6-route> ::= <IPV6_ADDRESS> <IPV6_PREFIX_LENGTH>
                [<multicast-source-ipv6-address>]
<mpls-route> ::= <MPLS> <MPLS_LABEL>
<mac-route> ::= <IEEE_MAC> ( <MAC_ADDRESS> )
<interface-route> ::= <INTERFACE> <INTERFACE_IDENTIFIER>

<multicast-source-ipv4-address> ::= <IPV4_ADDRESS>
                                   <IPV4_PREFIX_LENGTH>
<multicast-source-ipv6-address> ::= <IPV6_ADDRESS>

```

```

<IPV6_PREFIX_LENGTH>

<route-attributes> ::= [<ROUTE_PREFERENCE>] [<ROUTE_METRIC>]
                        [<LOCAL_ONLY>]
                        [<address-family-route-attributes>]

<address-family-route-attributes> ::= <ip-route-attributes> |
                                       <mpls-route-attributes> |
                                       <ethernet-route-attributes>

<ip-route-attributes> ::= [<as-path>] [<rpf-check-interface>]
<as-path> ::= (<as-path-segment-type> <as-list>) [<as-path> ...]
<as-path-segment-type> ::= <AS_SET> | <AS_SEQUENCE> |
                           <AS_CONFED_SEQUENCE> | <AS_CONFED_SET>
<as-list> ::= (<AS_NUMBER> ...) [<as-path>]

<rpf-check-interface> ::= <interface-list>

<mpls-route-attributes> ::= [<rpf-check-interface>]
<ethernet-route-attributes> ::= <>
<route-vendor-attributes> ::= <>

<nexthop-list> ::= <special-nexthop> |
                  ((<nexthop-list-member>) |
                   ([<nexthop-list-member> ... ] <nexthop-list> ))

<nexthop-list-member> ::= (<nexthop-chain> |
                           <nexthop-chain-identifier> )
                           [<nexthop-list-member-attributes>]
<nexthop-list-member-attributes> ::= [<PROTECTION_PREFERENCE>]
                                     [<LOAD_BALANCE_WEIGHT>]

<nexthop-chain> ::= (<nexthop> ...)
<nexthop-chain-identifier> ::= <NEXTHOP_NAME> | <NEXTHOP_ID>
<nexthop> ::= (<nexthop-identifier> | <EGRESS_IDENTIFIER> |
               (<nexthop-address>
                ([<ROUTING_TABLE_NAME>] | [<EGRESS_INTERFACE>]))) |
               (<tunnel-encap> [<EGRESS_INTERFACE>]) |
               <logical-tunnel> |
               <ROUTING_TABLE_NAME>)
               [<nexthop-attributes>]
               [<nexthop-vendor-attributes>]

<nexthop-identifier> ::= <NEXTHOP_NAME> | <NEXTHOP_ID>
<nexthop-address> ::= (<IPv4> <ipv4-address>) |
                      (<IPv6> <ipv6-address>) |
                      (<IEEE_MAC> <IEEE_MAC_ADDRESS>) |
                      (<ISO> <ISO_ADDRESS>)

```

```

<special-nexthop> ::= <DISCARD> | <DISCARD_WITH_ERROR> |
                      (<RECEIVE> [<COS_VALUE>] [<rate-limiter>])
<rate-limiter> ::= <>

<logical-tunnel> ::= <tunnel-type> <TUNNEL_NAME>
<tunnel-type> ::= <IP> | <MPLS> | <GRE> | <VxLAN> | <NVGRE>

<tunnel-encap> ::= (<IPv4> <ipv4-header>) |
                   (<IPv6> <ipv6-header>) |
                   (<MPLS> <mpls-header>) |
                   (<GRE> <gre-header>) |
                   (<VXLAN> <vxlan-header>) |
                   (<NVGRE> <nvgre-header>)

<ipv4-header> ::= <SOURCE_IPv4_ADDRESS> <DESTINATION_IPv4_ADDRESS>
                  <PROTOCOL> [<TTL>] [<DSCP>]

<ipv6-header> ::= <SOURCE_IPv6_ADDRESS> <DESTINATION_IPv6_ADDRESS>
                  <NEXT_HEADER> [<TRAFFIC_CLASS>]
                  [<FLOW_LABEL>] [<HOP_LIMIT>]

<mpls-header> ::= (<mpls-label-operation> ...)
<mpls-label-operation> ::= (<MPLS_PUSH> <MPLS_LABEL> [<S_BIT>]
                           [<TOS_VALUE>] [<TTL_VALUE>]) |
                           (<MPLS_POP> [<TTL_ACTION>])

<gre-header> ::= <GRE_IP_DESTINATION> <GRE_PROTOCOL_TYPE> [<GRE_KEY>]
<vxlan-header> ::= (<ipv4-header> | <ipv6-header>)
                   [<VXLAN_IDENTIFIER>]
<nvgre-header> ::= (<ipv4-header> | <ipv6-header>)
                   <VIRTUAL_SUBNET_ID>
                   [<FLOW_ID>]

<nexthop-attributes> ::= [<NEXTHOP_ADDRESS_FAMILY>]
                        [<nexthop-flags>]
<NEXTHOP_ADDRESS_FAMILY> ::= <IPv4> | <IPv6> | <ISO> | <IEEE MAC>
<nexthop-flags> ::= [<NO_DECREMENT_TTL>] [<NO_PROPAGATE_TTL>]
<nexthop-vendor-attributes> ::= <>

```

## 7. Using the RIB grammar

The RIB grammar is very generic and covers a variety of features. This section provides examples on using objects in the RIB grammar and examples to program certain use cases.

### 7.1. Using route preference and metric

Using route preference one can pre-install protection paths in the network. For example, if OSPF has a route preference of 10, then one can install a route with route preference of 20 to the same destination. The OSPF route will get precedence and will get installed in the FIB. When the OSPF route goes away (for any reason), the protection path will get installed in the FIB.

Route preference can also be used to prevent denial of service attacks by installing routes with the best preference, which either drops the offending traffic or routes it to some monitoring/analysis station. Since the routes are installed with the best preference, they will supersede any route installed by any other protocol.

Route metric is used to disambiguate between 2 or more routes to the same destination with the same preference and in the same route table. One usage of this is to install 2 routes, each with a different nexthop. The preferred nexthop is given a better metric than the other one. This results in traffic being forwarded to the preferred nexthop. If the preferred nexthop fails, then the RIB manager will automatically install a route to the other nexthop.

### 7.2. Using different nexthops types

The RIB grammar allows one to create a variety of nexthops. This section describes uses for certain types of nexthops.

#### 7.2.1. Tunnel nexthops

A tunnel nexthop points to a tunnel of some kind. Traffic that goes over the tunnel gets encapsulated with the tunnel encap. Tunnel nexthops are useful for abstracting out details of the network, by having the traffic seamlessly route between network edges.

#### 7.2.2. Replication lists

One can create a replication list for replication traffic to multiple destinations. The destinations, in turn, could be complex nexthops in themselves - at a level supported by the network device. Point to multipoint and broadcast are examples that involve replication.

A replication list (at the simplest level) can be represented as:

```
<nexthop-list> ::= <nexthop> [ <nexthop> ... ]
```

The above can be derived from the grammar as follows:

```
<nexthop-list> ::= <nexthop-list-member> [<nexthop-list-member> ...]
<nexthop-list> ::= <nexthop-chain> [<nexthop-chain> ...]
<nexthop-list> ::= <nexthop> [ <nexthop> ... ]
```

### 7.2.3. Weighted lists

A weighted list is used to load-balance traffic among a set of nexthops. A weighted list is very similar to a replication list, with the difference that each member nexthop MUST have a LOAD\_BALANCE\_WEIGHT associated with it.

A weighted list (at the simplest level) can be represented as:

```
<nexthop-list> ::= (<nexthop> <LOAD_BALANCE_WEIGHT>)
                  [(<nexthop> <LOAD_BALANCE_WEIGHT>)... ]
```

The above can be derived from the grammar as follows:

```
<nexthop-list> ::= <nexthop-list-member> [<nexthop-list-member> ...]
<nexthop-list> ::= (<nexthop-chain> <nexthop-list-member-attributes>)
                  [(<nexthop-chain>
                    <nexthop-list-member-attributes>) ...]
<nexthop-list> ::= (<nexthop-chain> <LOAD_BALANCE_WEIGHT>)
                  [(<nexthop-chain> <LOAD_BALANCE_WEIGHT>) ... ]
<network-list> ::= (<nexthop> <LOAD_BALANCE_WEIGHT>)
                  [(<nexthop> <LOAD_BALANCE_WEIGHT>)... ]
```

### 7.2.4. Protection lists

Protection lists are similar to weighted lists. A protection list specifies a set of primary nexthops and a set of backup nexthops. The <PROTECTION\_PREFERENCE> attribute indicates which nexthop is primary and which is backup.

A protection list can be represented as:

```
<nexthop-list> ::= (<nexthop> <PROTECTION_PREFERENCE>)
                  [(<nexthop> <PROTECTION_PREFERENCE>)... ]
```

A protection list can also be a weighted list. In other words, traffic can be load-balanced among the primary nexthops of a



protection list. In such a case, the list will look like:

```
<nexthop-list> ::= (<nexthop> <PROTECTION_PREFERENCE>
                    <LOAD_BALANCE_WEIGHT>)
                    [(<nexthop> <PROTECTION_PREFERENCE>
                     <LOAD_BALANCE_WEIGHT>)... ]
```

#### 7.2.5. Nexthop chains

A nexthop chain is a nexthop that puts one or more headers on an outgoing packet. One example is a Pseudowire - which is MPLS over some transport (MPLS or GRE for instance). Another example is VxLAN over IP. A nexthop chain allows an external entity to break up the programming of the nexthop into independent pieces - one per encapsulation.

A simple example of MPLS over GRE can be represented as:

```
<nexthop-list> ::= (<MPLS> <mpls-header>) (<GRE> <gre-header>)
```

The above can be derived from the grammar as follows:

```
<nexthop-list> ::= <nexthop-list-member> [<nexthop-list-member> ...]
<nexthop-list> ::= <nexthop-chain>
<nexthop-list> ::= <nexthop> [ <nexthop> ... ]
<nexthop-list> ::= <tunnel-encap> (<nexthop> [ <nexthop> ...])
<nexthop-list> ::= <tunnel-encap> (<tunnel-encap>)
<nexthop-list> ::= (<MPLS> <mpls-header>) (<GRE> <gre-header>)
```

#### 7.2.6. Lists of lists

Lists of lists is a complex construct. One example of usage of such a construct is to replicate traffic to multiple destinations, with high availability. In other words, for each destination you have a primary and backup nexthop (replication list) to ensure there is no traffic drop in case of a failure. So the outer list is a list of destinations and the inner lists are replication lists of primary/backup nexthops.

#### 7.3. Performing multicast

IP multicast involves matching a packet on (S, G) or (\*, G), where both S (source) and G (group) are IP prefixes. Following the match, the packet is replicated to one or more recipients. How the recipients subscribe to the multicast group is outside the scope of this document.

In PIM-based multicast, the packets are IP forwarded on an IP multicast tree. The downstream nodes on each point in the multicast tree is one or more IP addresses. These can be represented as a replication list ( Section 7.2.2 ).

In MPLS-based multicast, the packets are forwarded on a point to multipoint (P2MP) label-switched path (LSP). The nexthop for a P2MP LSP can be represented in the nexthop grammar as a <logical-tunnel> (P2MP LSP identifier) or a replication list ( Section 7.2.2 ) of <tunnel-encap>, with each tunnel encap representing a single mpls downstream nexthop.

#### 7.4. Solving optimized exit control

In case of optimized exit control, a controller wants to control the edge device (and optionally control the outgoing interface on that edge device) that is used by a server to send traffic out. This can be easily achieved by having the controller program the edge router (Eg. 192.0.2.10) and the server along the following lines:

Server:

```
<route> ::= <routing-table-name> <match> (<edge-router>
                                         <edge-router-interface>)
<route> ::= <routing-table-name> <198.51.100.1/16>
            (<MPLS> <mpls-header>)
            (<GRE> <gre-header>)

<route> ::= <routing-table-name> <198.51.100.1/16>
            (<MPLS_PUSH> <100>)
            (<GRE> <192.0.2.10> <GRE_PROTOCOL_MPLS>)
```

Edge Router:

```
<route> ::= <mpls-routing-table> <mpls-route> <nexthop>
<route> ::= <mpls-routing-table> (<MPLS> <100>) <interface-10>
```

In the above case, the label 100 identifies the egress interface on the edge router.

## 8. RIB operations at scale

This section discusses the scale requirements for a RIB data-model. The RIB data-model should be able to handle large scale of operations, to enable deployment of RIB applications in large networks.

### 8.1. RIB reads

Bulking (grouping of multiple objects in a single message) MUST be supported when a network device sends RIB data to an external entity.

### 8.2. RIB writes

Bulking (grouping of multiple write operations in a single message) MUST be supported when an external entity wants to write to the RIB. The response from the network device MUST include a return-code for each write operation in the bulk message.

### 8.3. RIB events and notifications

There can be cases where a single network event results in multiple events and/or notifications from the network device to an external entity. On the other hand, due to timing of multiple things happening at the same time, a network device might have to send multiple events and/or notifications to an external entity. The network device originated event/notification message MUST support bulking of multiple events and notifications in a single message.

## 9. Security Considerations

All interactions between a RIB manager and an external entity MUST be authenticated. The RIB manager MUST protect itself against a denial of service attack by a rouge external entity, by throttling request processing. A RIB manager MUST enforce limits on how much data can be programmed by an external entity and return error when such a limit is reached.

The RIB manager MUST expose a data-model that it implements. An external agent MUST send requests to the RIB manager that comply with the supported data-model. The data-model MUST specify the behavior of the RIB manager on handling of unsupported data requests.

## 10. IANA Considerations

This document does not generate any considerations for IANA.

## 11. Acknowledgements

The authors would like to thank Alia Atlas, Edward Crabbe, Hariharan Ananthakrishnan, Jeff Haas and Ina Minei on their comments and suggestions on this draft. The following people contributed to the

design of the RIB model as part of the I2RS Interim meeting in April 2013 - Wes George, Chris Liljenstolpe, Jeff Tantsura, Sriganesh Kini, Susan Hares, Fabian Schneider and Nitin Bahadur.

## 12. References

### 12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

### 12.2. Informative References

- [I-D.atlas-i2rs-problem-statement]  
Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Problem Statement", draft-atlas-i2rs-problem-statement-01 (work in progress), July 2013.
- [I-D.hares-i2rs-use-case-vn-vc]  
Hares, S., "Use Cases for Virtual Connections on Demand (VCoD) and Virtual Network on Demand using Interface to Routing System", draft-hares-i2rs-use-case-vn-vc-00 (work in progress), February 2013.
- [I-D.white-i2rs-use-case]  
White, R., Hares, S., and R. Fernando, "Use Cases for an Interface to the Routing System", draft-white-i2rs-use-case-00 (work in progress), February 2013.
- [RFC3443] Agarwal, P. and B. Akyol, "Time To Live (TTL) Processing in Multi-Protocol Label Switching (MPLS) Networks", RFC 3443, January 2003.
- [RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.
- [RFC4915] Psenak, P., Mirtorabi, S., Roy, A., Nguyen, L., and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF", RFC 4915, June 2007.
- [RFC5065] Traina, P., McPherson, D., and J. Scudder, "Autonomous System Confederations for BGP", RFC 5065, August 2007.
- [RFC5120] Przygienda, T., Shen, N., and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to

Intermediate Systems (IS-IS)", RFC 5120, February 2008.

[RFC5511] Farrel, A., "Routing Backus-Naur Form (RBNF): A Syntax Used to Form Encoding Rules in Various Routing Protocol Specifications", RFC 5511, April 2009.

#### Authors' Addresses

Nitin Bahadur (editor)  
Juniper Networks, Inc.  
1194 N. Mathilda Avenue  
Sunnyvale, CA 94089  
US

Phone: +1 408 745 2000  
Email: [nitinb@juniper.net](mailto:nitinb@juniper.net)  
URI: [www.juniper.net](http://www.juniper.net)

Ron Folkes (editor)  
Juniper Networks, Inc.  
1194 N. Mathilda Avenue  
Sunnyvale, CA 94089  
US

Phone: +1 408 745 2000  
Email: [ronf@juniper.net](mailto:ronf@juniper.net)  
URI: [www.juniper.net](http://www.juniper.net)

Sriganesh Kini  
Ericsson

Email: [sriganesh.kini@ericsson.com](mailto:sriganesh.kini@ericsson.com)

Jan Medved  
Cisco

Email: [jmedved@cisco.com](mailto:jmedved@cisco.com)

