

Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: August 16, 2013

S. Amante  
Level 3 Communications, Inc.  
J. Medved  
S. Previdi  
Cisco Systems, Inc.  
T. Nadeau  
Juniper Networks  
February 12, 2013

Topology API Use Cases  
draft-amante-i2rs-topology-use-cases-00

Abstract

This document describes use cases for gathering routing, forwarding and policy information, (hereafter referred to as topology information), about the network. It describes several applications that need to view the topology of the underlying physical or logical network. This document further demonstrates a need for a "Topology Manager" and related functions that collect topology data from network elements and other data sources, coalesces the collected data into a coherent view of the overall network topology, and normalizes the network topology view for use by clients -- namely, applications that consume topology information.

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].

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 16, 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.

## Table of Contents

1. Introduction . . . . .	3
1.1. Statistics Collection . . . . .	5
1.2. Inventory Collection . . . . .	5
1.3. Requirements Language . . . . .	6
2. Terminology . . . . .	6
3. Orchestration, Collection & Presentation Framework . . . . .	7
3.1. Overview . . . . .	7
3.2. Topology Manager . . . . .	8
3.3. Policy Manager . . . . .	10
3.4. Orchestration Manager . . . . .	10
4. Use Cases . . . . .	12
4.1. Virtualized Views of the Network . . . . .	12
4.1.1. Capacity Planning and Traffic Engineering . . . . .	12
4.1.2. Services Provisioning . . . . .	15
4.1.3. Troubleshooting & Monitoring . . . . .	15
4.2. Path Computation Element (PCE) . . . . .	16
4.3. ALTO Server . . . . .	17
5. Acknowledgements . . . . .	18
6. IANA Considerations . . . . .	18
7. Security Considerations . . . . .	19
8. References . . . . .	19
8.1. Normative References . . . . .	19
8.2. Informative References . . . . .	19
Authors' Addresses . . . . .	20

## 1. Introduction

In today's networks, a variety of applications, such as Traffic Engineering, Capacity Planning, Security Auditing or Services Provisioning (for example, Virtual Private Networks), have a common need to acquire and consume network topology information. Unfortunately, all of these applications are (typically) vertically integrated: each uses its own proprietary normalized view of the network and proprietary data collectors, interpreters and adapters, which speak a variety of protocols, (SNMP, CLI, SQL, etc.) directly to network elements and to back-office systems. While some of the topological information can be distributed using routing protocols, unfortunately it is not desirable for some of these applications to understand or participate in routing protocols.

This approach is incredibly inefficient for several reasons. First, developers must write duplicate 'network discovery' functions, which then become challenging to maintain over time, particularly as/when new equipment are first introduced to the network. Second, since there is no common "vocabulary" to describe various components in the network, such as physical links, logical links, or IP prefixes, each application has its own data model. To solve this, some solutions have distributed this information in the normalized form of routing distribution. However, this information still does not contain "inactive" topological information, thus not containing information considered to be part of a network's inventory.

These limitations lead to applications being unable to easily exchange information with each other. For example, applications cannot share changes with each other that are (to be) applied to the physical and/or logical network, such as installation of new physical links, or deployment of security ACL's. Each application must frequently poll network elements and other data sources to ensure that it has a consistent representation of the network so that it can carry out its particular domain-specific tasks. In other cases, applications that cannot speak routing protocols must use proprietary CLI or other management interfaces which represent the topological information in non-standard formats or worse, semantic models.

Overall, the software architecture described above at best results in incredibly inefficient use of both software developer resources and network resources, and at worst, it results in some applications simply not having access to this information.

Figure 1 is an illustration of how individual applications collect data from the underlying network. Applications retrieve inventory, network topology, state and statistics information by communicating directly with underlying Network Elements as well as with

intermediary proxies of the information. In addition, applications transmit changes required of a Network Element's configuration and/or state directly to individual Network Elements, (most commonly using CLI or Netconf). It is important to note that the "data models" or semantics of this information contained within Network Elements are largely proprietary with respect to most configuration and state information, hence why a proprietary CLI is often the only choice to reflect changes in a NE's configuration or state. This remains the case even when standards-based mechanisms such as Netconf are used which provide a standard syntax model, but still often lack due to the proprietary semantics associated with the internal representation of the information.

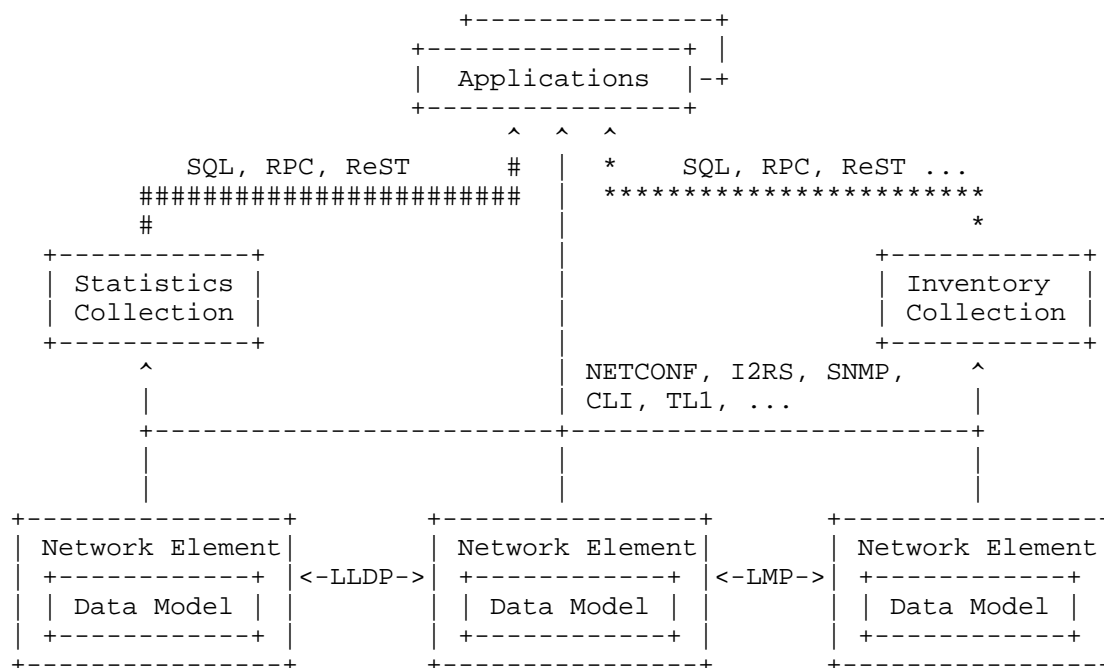


Figure 1: Applications getting topology data

Figure 1 shows how current management interfaces such as NETCONF, SNMP, CLI, etc. are used to transmit or receive information to/from various Network Elements. The figure also shows that protocols such as LLDP and LMP participate in topology discovery, specifically to discover adjacent network elements.

The following sections describe the "Statistics Collection" and "Inventory Collection" functions.

### 1.1. Statistics Collection

In Figure 1, "Statistics Collection" is a dedicated infrastructure that collects statistics from Network Elements. It periodically polls Network Elements (for example, every 5-minutes) for octets transferred per interface, per LSP, etc. Collected statistics are stored and collated, (for example, to provide hourly, daily, weekly 95th-percentile figures), within the statistics data warehouse. Applications typically query the statistics data warehouse rather than poll Network Elements directly to get the appropriate set of link utilization figures for their analysis.

### 1.2. Inventory Collection

"Inventory Collection" is a network function responsible for collecting network element component and state (i.e.: interface up/down, SFP/XFP optics inserted into physical port, etc.) information directly from network elements, as well as storing inventory information about physical network assets that are not retrievable from network elements, (hereafter referred to as a inventory asset database). Inventory Collection from network elements commonly use SNMP and CLI to acquire inventory information. The information housed in the Inventory Manager is retrieved by applications using a variety of protocols: SQL, RPC, etc. Inventory information, retrieved from Network Elements, is updated in the Inventory Collection system on a periodic basis to reflect changes in the physical and/or logical network assets. The polling interval to retrieve updated information is varied depending on scaling constraints of the Inventory Collection systems and expected intervals at which changes to the physical and/or logical assets are expected to occur.

Examples of changes in network inventory that need be learned by the Inventory Collection function are as follows:

- o Discovery of new Network Elements. These elements may or may not be actively used in the network (i.e.: provisioned but not yet activated).
- o Insertion or removal of line cards or other modules, i.e.: optics modules, during service or equipment provisioning.
- o Changes made to a specific Network Element through a management interface by a field technician.
- o Indication of an NE's physical location and associated cable run list, at the time of installation.

- o Insertion or removal of cables that result in dynamic discovery of a new or lost adjacent neighbor, etc.

### 1.3. 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. Terminology

The following briefly defines some of the terminology used within this document.

**Inventory Manager:** Describes a function of collecting network element inventory and state information directly from network elements, and potentially associated offline inventory databases, via standards-based data models. Components contained in this super set might be visible or invisible to a specific network layer, i.e.: a physical link is visible within the IGP, however the Layer-2 switch through which the physical link traverses is unknown to the Layer-3 IGP. .

**Policy Manager:** Describes a function of attaching metadata to network components/attributes. Such metadata is likely to include security, routing, L2 VLAN ID, IP numbering, etc. policies that enable the Topology Manager to: a) assemble a normalized view of the network for clients to access; b) allow clients (or, upper-layer applications) access to information collected from various network layers and/or network components, etc. The Policy Manager function may be a sub-component of the Topology Manager or it may be a standalone. This will be determined as the work with I2RS evolves.

**Topology Manager:** Network components (inventory, etc.) are retrieved from the Inventory Manager and synthesized with information from the Policy Manager into cohesive, normalized views of network layers. The Topology Manager exposes normalized views of the network via standards-based data models to Clients, or higher-layer applications, to act upon in a read-only and/or read-write fashion. The Topology Manager may also push information back into the Inventory Manager and/or Network Elements to execute changes to the network's behavior, configuration or state.

**Orchestration Manager:** Describes a function of stitching together resources (i.e.: compute, storage) and/or services with the network or vice-versa. The Orchestration Manager relies on the capabilities provided by the other "Managers" listed above in order to realize a complete service.

**Normalized Topology Data Model:** A data model that is constructed and represented using an open, standards-based model that is consistent between implementations.

**Data Model Abstraction:** The notion that one is able to represent the same set of elements in a data model at different levels of "focus" in order to limit the amount of information exchanged in order to convey this information.

**Multi-Layer Topology:** Topology is commonly referred to using the OSI protocol layering model. For example, Layer 3 represents routed topologies that typically use IPv4 or IPv6 addresses. It is envisioned that, eventually, multiple layers of the network may be represented in a single, normalized view of the network to certain applications, (i.e.: Capacity Planning, Traffic Engineering, etc.)

**Network Element (NE):** refers to a network device that typically is addressable (but not always), or a host. It is sometimes referred to as a 'Node'.

**Links:** Every NE contains at least 1 link. These are used to connect the NE to other NEs in the network. Links may be in a variety of states including up, down, administratively down, internally testing, or dormant. Links are often synonymous with network ports on NEs.

### 3. Orchestration, Collection & Presentation Framework

#### 3.1. Overview

Section 1 demonstrates the need for a network function that would provide a common, standard-based topology view to applications. Such topology collection/management/presentation function would be a part wider framework, that would also include policy management and orchestration. The framework is shown in Figure 2.

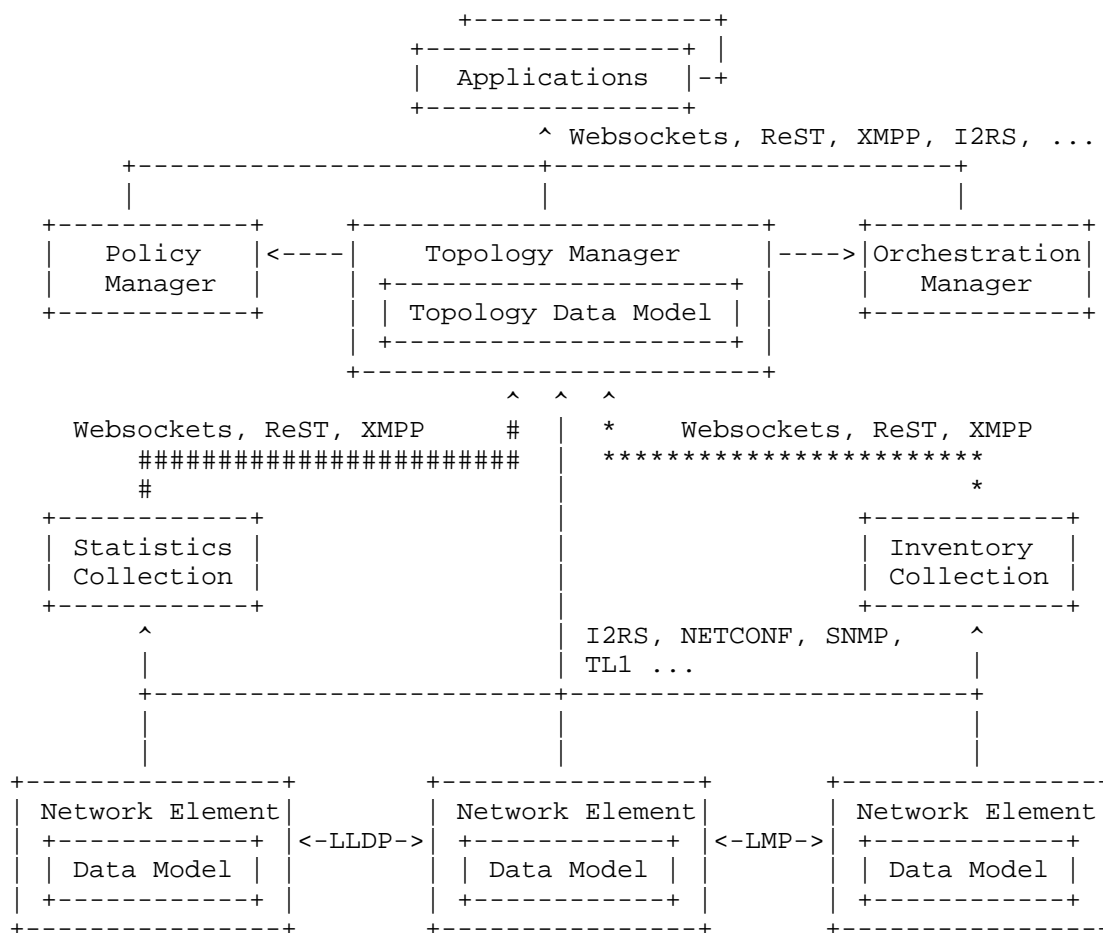


Figure 2: Topology Manager

The following sections describe in detail the Topology Manager, Policy Manager and Orchestration Manager functions.

### 3.2. Topology Manager

The Topology Manager is responsible for retrieving topological information from the network via a variety of sources. The first most obvious source is the "live" IGP or an equivalent mechanism. "Live" IGP provides information about links that are components of the active topology, in other words links that are present in the Link State Database (LSDB) and are eligible for forwarding. The second source of topology information is the Inventory Collection system, which provides information for network components not visible

within the IGP's LSDB, (i.e.: links or nodes, or properties of those links or nodes, at lower layers of the network). The third source of topology information is the Statistics Collection system, which provides traffic information (traffic demands, link utilizations, etc.).

The Topology Manager will synthesize retrieved information into cohesive, abstracted views of the network using a standards-based, normalized topology data model. The Topology Manager can then expose these data models to Clients, or higher-layer applications using a northbound interface, which would be a protocol/API commonly used by higher-layer applications to retrieve and update information. Examples of such protocols are ReST, Websockets, or XMPP. Topology Manager's clients would be able to act upon the information in a read-only and/or read-write fashion, (based on policies defined within the Policy Manager).

It is envisioned that the Topology Manager will ultimately contain topology information for multiple layers of the network: Transport, Ethernet and IP/MPLS as well as multiple (IGP) areas and/or multiple Autonomous Systems (ASes). This allows the Topology Manager to stitch together a holistic view of several layers of the network, which is an important requirement, particularly for upper-layer Traffic Engineering, Capacity Planning and Provisioning Clients (applications) used to design, augment and optimize IP/MPLS networks that require knowledge of underlying Shared Risk Link Groups (SRLG) within the Transport and/or Ethernet layers of the network.

The Topology Manager must have the ability to discover and communicate with network elements who are not only active and visible within the Link State Database (LSDB) of an active IGP, but also network elements who are active, but invisible to a LSDB (e.g.: L2 Ethernet switches, ROADMs, etc.) that are part of the underlying Transport network. This requirement will influence the choice of protocols needed by the Topology Manager to communicate to/from network elements at the various network layers.

It is also important to recognize that the Topology Manager will be gleaning not only (relatively) static inventory information from the Inventory Manager, i.e.: what linecards, interface types, etc. are actively inserted into network elements, but also dynamic inventory information, as well. With respect to the latter, network elements are expected to rely on various Link Layer Discovery Protocols (i.e.: LLDP, LMP, etc.) that will aid in automatically identifying an adjacent node, port, etc. at the far-side of a link. This information is then pushed to or pulled by the Topology Manager in order for it to have an accurate representation of the physical topology of the network.

### 3.3. Policy Manager

The Policy Manager is the function used to enforce and program policies applicable to network component/attribute data. Policy enforcement is a network-wide function that can be consumed by various network elements and services including the Inventory Manager, Topology Manager or other network elements. Such policies are likely to encompass the following.

- o Logical Identifier Numbering Policies
  - \* Correlation of IP prefix to link based on type of link (P-P, P-PE, PE-CE, etc.)
  - \* Correlation of IP Prefix to IGP Area
  - \* Layer-2 VLAN ID assignments, etc.
- o Routing Configuration Policies
  - \* OSPF Area or IS-IS Net-ID to Node (Type) Correlation
  - \* BGP routing policies, i.e.: nodes designated for injection of aggregate routes, max-prefix policies, AFI/SAFI to node correlation, etc.
- o Security Policies
  - \* Access Control Lists
  - \* Rate-limiting
- o Network Component/Attribute Data Access Policies. Client's (upper-layer application) read-only or read-write access to Network Components/Attributes contained in the "Inventory Manager" as well as Policies contained within the "Policy Manager" itself.

The Policy Manager function may be a sub-component of the Topology or Orchestration Manager or it may be a standalone. This will be determined as the work with I2RS evolves.

### 3.4. Orchestration Manager

The Orchestration Manager provides the ability to stitch together resources (i.e.: compute, storage) and/or services with the network or vice-versa. Examples of 'generic' services may include the following:

- o Application-specific Load Balancing
- o Application-specific Network (Bandwidth) Optimization
- o Application or End-User specific Class-of-Service
- o Application or End-User specific Network Access Control

The above services could then enable coupling of resources with the network to realize the following:

- o Network Optimization: Creation and Migration of Virtual Machines (VM's) so they are adjacent to storage in the same DataCenter.
- o Network Access Control: Coupling of available (generic) compute nodes within the appropriate point of the data-path to perform firewall, NAT, etc. functions on data traffic.

The Orchestration Manager is expected to exchange data models with the Topology Manager, Policy Manager and Inventory Manager functions. In addition, the Orchestration Manager is expected to support publish and subscribe capabilities to those functions, as well as to Clients, to enable scalability with respect to event notifications.

The Orchestration Manager may receive requests from Clients (applications) for immediate access to specific network resources. However, Clients may request to schedule future appointments to reserve appropriate network resources when, for example, a special event is scheduled to start and end.

Finally, the Orchestration Manager should have the flexibility to determine what network layer(s) may be able to satisfy a given Client's request, based on constraints received from the Client as well as those constraints learned from the Policy and/or Topology Manager functions. This could allow the Orchestration Manager to, for example, satisfy a given service request for a given Client using the optical network (via OTN service) if there is insufficient IP/MPLS capacity at the specific moment the Client's request is received.

The operational model is shown in the following figure.

TBD.

Figure 3: Overall Reference Model

## 4. Use Cases

### 4.1. Virtualized Views of the Network

#### 4.1.1. Capacity Planning and Traffic Engineering

When performing Traffic Engineering and/or Capacity Planning of an IP/MPLS network, it is important to account for SRLG's that exist within the underlying physical, optical and Ethernet networks. Currently, it's quite common to create and/or take "snapshots", at infrequent intervals, that comprise the inventory data of the underlying physical and optical layer networks. This inventory data then needs to be massaged or normalized to conform to the data import requirements of sometimes separate Traffic Engineering and/or Capacity Planning tools. This process is error-prone and inefficient, particularly as the underlying network inventory information changes due to introduction of, for example, new network element makes or models, linecards, capabilities, etc. at the optical and/or Ethernet layers of the underlying network.

This is inefficient with respect to the time and expense consumed by software developer, Capacity Planning and Traffic Engineering resources to normalize and sanity check underlying network inventory information, before it can be consumed by IP/MPLS Capacity Planning and Traffic Engineering applications. Due to this inefficiency, the underlying physical network inventory information, (containing SRLG and corresponding critical network asset information), used by the IP/MPLS Capacity Planning and TE applications is not updated frequently, thus exposing the network to, at minimum, inefficient utilization and, at worst, critical impairments.

An Inventory Manager function is required that will, first, extract inventory information from network elements -- and potentially associated offline inventory databases to acquire physical cross-connects and other information that is not available directly from network elements -- at the physical, optical, Ethernet and IP/MPLS layers of the network via standards-based data models. Data models and associated vocabulary will be required to represent not only components inside or directly connected to network elements, but also to represent components of a physical layer path (i.e.: cross-connect panels, etc.) The aforementioned inventory will comprise the complete set of inactive and active network components.

A Statistics Collection Function is also required. As stated above, it will collect utilization statistics from Network Elements, archive and aggregate them in a statistics data warehouse. Selected statistics and other dynamic data may be distributed through IGP routing protocols ([I-D.previdi-isis-te-metric-extensions] and

[I-D.ietf-ospf-te-metric-extensions]) and then collected at the Statistics Collection Function via BGP-LS ([I-D.ietf-idr-ls-distribution]). Summaries of these figures then need to be exposed in normalized data models to the Topology Manager so it can easily acquire historical link and LSP utilization figures that can be used to, for example, build trended utilization models to forecast expected changes to the physical and/or logical network components to accommodate network growth.

The Topology Manager function may then augment the Inventory Manager information by communicating directly with Network Elements to reveal the IGP-based view of the active topology of the network. This will allow the Topology Manager to include dynamic information from the IGP, such as Available Bandwidth, Reserved Bandwidth, etc. Traffic Engineering (TE) attributes associated with links, contained with the Traffic Engineering Database (TED) on Network Elements.

It is important to recognize that extracting topology information from the network solely via an IGP, (such as IS-IS TE or OSPF TE), is inadequate for this use case. First, IGP's only expose the active components (e.g. vertices of the SPF tree) of the IP network; unfortunately, they are not aware of "hidden" or inactive interfaces within IP/MPLS network elements, (e.g.: unused linecards or unused ports), or components that reside at a lower layer than IP/MPLS, e.g. Ethernet switches, Optical transport systems, etc. This occurs frequently during the course of maintenance, augment and optimization activities on the network. Second, IGP's only convey SRLG information that have been first applied within the router's configurations, either manually or programatically. As mentioned previously, this SRLG information in the IP/MPLS network is subject to being infrequently updated and, as a result, may inadequately account for critical, underlying network fate sharing properties that are necessary to properly design resilient circuits and/or paths through the network.

In this use case, the Inventory Manager will need to be capable of using a variety of existing protocols such as: NETCONF, CLI, SNMP, TL1, etc. depending on the capabilities of the network elements. The Topology Manager will need to be capable of communicating via an IGP from a (set of) Network Elements. It is important to consider that to acquire topology information from Network Elements will require read-only access to the IGP. However, the end result of the computations performed by the Capacity Planning Client may require changes to various IGP attributes, (e.g.: IGP metrics, TE link-colors, etc.) These may be applied directly by devising a new capability to either: a) inject information into the IGP that overrides the same information injected by the originating Network Element; or, b) allowing the Topology and/or Inventory Manager the

ability to write changes to the Network Element's configuration in order to have it adjust the appropriate IGP attribute(s) and re-flood them throughout the IGP. It would be desirable to have a single mechanism (data model or protocol) that allows the Topology Manager to read and write IGP attributes.

Once the Topology Manager function has assembled a normalized view of the topology and synthesized associated metadata with each component of the topology (link type, link properties, statistics, intra-layer relationships, etc.), it can then expose this information via its northbound API to Clients. In this use case that means Capacity Planning and Traffic Engineering applications, which are not required to know innate details of individual network elements, but do require generalized information about the node and links that comprise the network, e.g.: links used to interconnect nodes, SRLG information (from the underlying network), utilization rates of each link over some period of time, etc. In this case, it is important that any Client that understands both the web services API and the normalized data model can communicate with the Topology Manager in order to understand the network topology information that was provided by network elements from potentially different vendors, all of which likely represent that topology information internally using different models. If the Client had gone directly to the network elements themselves, it would have to translate and then normalize these different representations for itself. However, in this case, the Topology Manager has done that for it.

When this information is consumed by the Traffic Engineering application, it may run a variety of CSPF algorithms the result of which is likely a list of RSVP LSP's that need to be (re-)established, or torn down, in the network to globally optimize the packing efficiency of physical links throughout the network. The end result of the Traffic Engineering application is "pushing" out to the Topology Manager, via a standard data model to be defined here, a list of RSVP LSP's and their associated characteristics, (i.e.: head and tail-end LSR's, bandwidth, priority, preemption, etc.). The Topology Manager then would consume this information and carry out those instructions by speaking directly to network elements, perhaps via PCEP Extensions for Stateful PCE [I-D.ietf-pce-stateful-pce], which in turn initiates RSVP signaling through the network to establish the LSP's.

After this information is consumed by the Capacity Planning application, it may run a variety of algorithms the result of which is a list of new inventory that is required to be purchased (or, redeployed) as well as associated work orders for field technicians to augment the network for expected growth. It would be ideal if this information was also "pushed" back into the Topology and, in

turn, Inventory Manager as "inactive" links and/or nodes, so that as new equipment is installed it can be automatically correlated with original design and work order packages associated with that augment.

#### 4.1.2. Services Provisioning

Beyond Capacity Planning and Traffic Engineering applications, having a normalized view of just the IP/MPLS layer of the network is still very important for other mission critical applications such as Security Auditing and IP/MPLS Services Provisioning, (e.g.: L2VPN, L3VPN, etc.). With respect to the latter, these types of applications should not need a detailed understanding of, for example, SRLG information, assuming that the underlying MPLS Tunnel LSP's are known to account for the resiliency requirements of all services that ride over them. Nonetheless, for both types of applications it is critical that they have a common and up-to-date normalized view of the IP/MPLS network in order to easily instantiate new services at the appropriate places in the network, in the case of VPN services, or validate that ACL's are configured properly to protect associated routing, signaling and management protocols on the network, with respect to Security Auditing.

For this use case, what is most commonly needed by a VPN Service Provisioning application is as follows. First, Service PE's need to be identified in all markets/cities where the customer has identified they want service. Next, does their exist one, or more, Services PE's in each city with connectivity to the access network(s), e.g.: SONET/TDM, used to deliver the PE-CE tail circuits to the Service's PE. Finally, does the Services PE have available capacity on both the PE-CE access interface and its uplinks to terminate the tail circuit? If this were to be generalized, this would be considered an Resource Selection function. Namely, the VPN Provisioning application would iteratively query the Topology Manager to narrow down the scope of resources to the set of Services PE's with the appropriate uplink bandwidth and access circuit capability plus capacity to realize the requested VPN service. Once the VPN Provisioning application has a candidate list of resources it then requests the Topology Manager to go about configuring the Services PE's and associated access circuits to realize the customer's VPN service.

#### 4.1.3. Troubleshooting & Monitoring

Once the Topology Manager has a normalized view of several layers of the network, it's then possible to more easily expose a richer set of data to network operators when performing diagnosis, troubleshooting and repairs on the network. Specifically, there is a need to (rapidly) assemble a current, accurate and comprehensive network diagram of a L2VPN or L3VPN service for a particular customer when

either: a) attempting to diagnose a service fault/error; or, b) attempting to augment the customer's existing service. Information that may be assembled into a comprehensive picture could include physical and logical components related specifically to that customer's service, i.e.: VLAN's or channels used by the PE-CE access circuits, CoS policies, historical PE-CE circuit utilization, etc. The Topology Manager would assemble this information, on behalf of each of the network elements and other data sources in and associated with the network, and could present this information in a vendor-independent data model to applications to be displayed allowing the operator (or, potentially, the customer through a SP's Web portal) to visualize the information.

#### 4.2. Path Computation Element (PCE)

As described in [RFC4655] a PCE can be used to compute MPLS-TE paths within a "domain" (such as an IGP area) or across multiple domains (such as a multi-area AS, or multiple ASes).

- o Within a single area, the PCE offers enhanced computational power that may not be available on individual routers, sophisticated policy control and algorithms, and coordination of computation across the whole area.
- o If a router wants to compute a MPLS-TE path across IGP areas its own TED lacks visibility of the complete topology. That means that the router cannot determine the end-to-end path, and cannot even select the right exit router (Area Border Router - ABR) for an optimal path. This is an issue for large-scale networks that need to segment their core networks into distinct areas, but which still want to take advantage of MPLS-TE.

The PCE presents a computation server that may have visibility into more than one IGP area or AS, or may cooperate with other PCEs to perform distributed path computation. The PCE needs access to the topology and the Traffic Engineering Database (TED) for the area(s) it serves, but [RFC4655] does not describe how this is achieved. Many implementations make the PCE a passive participant in the IGP so that it can learn the latest state of the network, but this may be sub-optimal when the network is subject to a high degree of churn, or when the PCE is responsible for multiple areas.

The following figure shows how a PCE can get its TED information using a Topology Server.

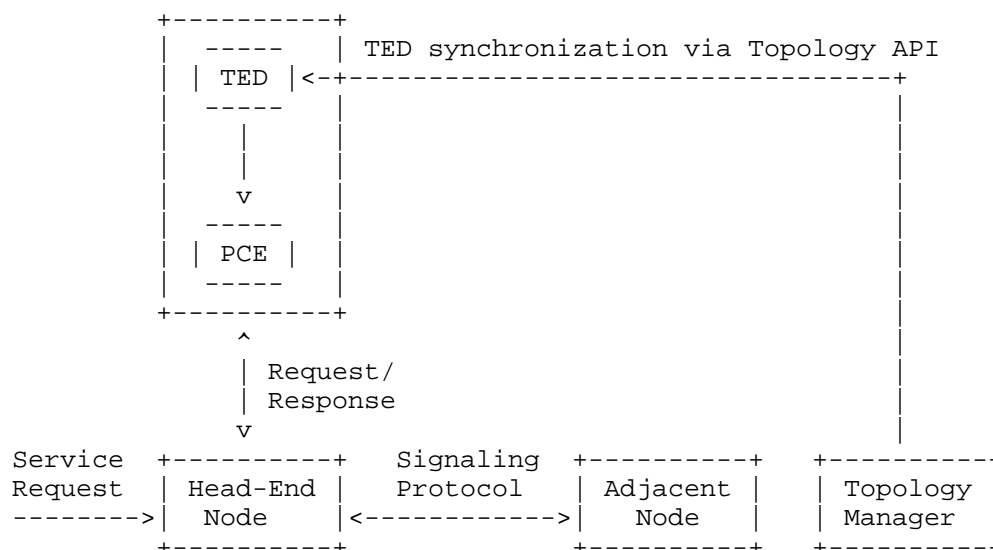


Figure 4: Topology use case: Path Computation Element

#### 4.3. ALTO Server

An ALTO Server [RFC5693] is an entity that generates an abstracted network topology and provides it to network-aware applications over a web service based API.

Example applications are Content Delivery Network (CDNs), peer-to-peer clouds/swarms, as well as inter-layer optimization cases such as mobile network willing to understand the congestion level of underneath backhaul infrastructure.

ALTO mechanisms are based on "Maps" that contain an abstracted version of the topology. Such Maps are built by the ALTO server or made available to the ALTO server by a Topology Manager. The content of Maps are multiple: a mapping list where each prefix is mapped into a Partition Identifier (called PID) and the cost matrix (representing the distance) between PIDs. For more details, see [I-D.ietf-alto-protocol].

ALTO abstract network topologies (represented in the Maps) can be generated in multiple ways among which the Topology Manager provides the abstracted topology to the ALTO server so that the ALTO server is capable of serving applications. ALTO Maps may represent the whole network infrastructure and are not limited to a specific layer. E.g.: the cost matrix (called the Cost Map) can represent the IP/MPLS layer path costs as well as integrating the optical cost.

The generation would typically be based on policies and rules set by the operator. All the relevant information such as Nodes, Links, Prefixes, TE paths (LSPs/Tunnels), etc. is required so for the ALTO server to have an exhaustive and consistent view of the infrastructure.

Typically, a Topology Manager would aggregate all the necessary information and would produce ALTO maps. Mechanisms through which a Topology Manager acquires topology information include interaction with the IGP and the use of BGP-LS extension.

The mechanism defined in this document provides a single interface through which an ALTO Server can retrieve all the necessary prefix and network topology data from the underlying network (i.e.: the Topology Manager). Note an ALTO Server can use other mechanisms to get network data, for example, peering with multiple IGP and BGP Speakers.

The following figure shows how an ALTO Server can get network topology information from the underlying network using the Topology API.

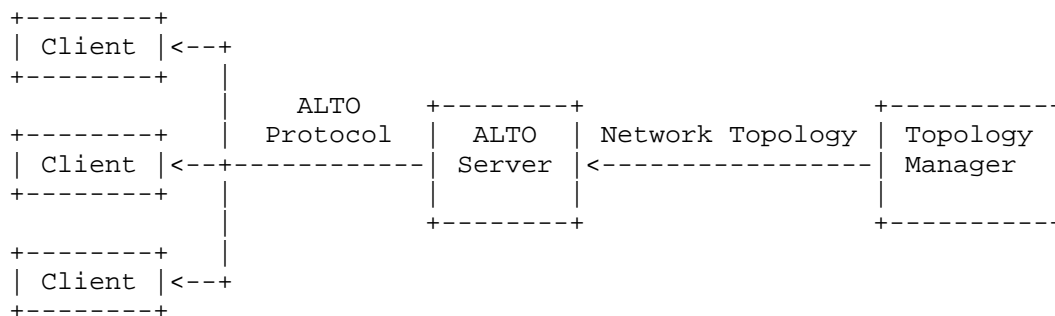


Figure 5: Topology use case: ALTO Server

## 5. Acknowledgements

The authors wish to thank Alia Atlas, Dave Ward and Hannes Gredler for their valuable contributions and feedback to this draft.

## 6. IANA Considerations

This memo includes no request to IANA.

## 7. Security Considerations

At the moment, the Use Cases covered in this document apply specifically to a single Service Provider or Enterprise network. Therefore, network administrations should take appropriate precautions to ensure appropriate access controls exist so that only internal applications and end-users have physical or logical access to the Topology Manager. This should be similar to precautions that are already taken by Network Administrators to secure their existing Network Management, OSS and BSS systems.

As this work evolves, it will be important to determine the appropriate granularity of access controls in terms of what individuals or groups may have read and/or write access to various types of information contained with the Topology Manager. It would be ideal, if these access control mechanisms were centralized within the Topology Manager itself.

## 8. References

### 8.1. Normative References

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

### 8.2. Informative References

- [I-D.ietf-alto-protocol]  
Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol", draft-ietf-alto-protocol-13 (work in progress), September 2012.
- [I-D.ietf-idr-ls-distribution]  
Gredler, H., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and TE Information using BGP", draft-ietf-idr-ls-distribution-01 (work in progress), October 2012.
- [I-D.ietf-ospf-te-metric-extensions]  
Giacalone, S., Ward, D., Drake, J., Atlas, A., and S. Previdi, "OSPF Traffic Engineering (TE) Metric Extensions", draft-ietf-ospf-te-metric-extensions-02 (work in progress), December 2012.
- [I-D.ietf-pce-stateful-pce]  
Crabbe, E., Medved, J., Minei, I., and R. Varga, "PCEP Extensions for Stateful PCE",

draft-ietf-pce-stateful-pce-02 (work in progress),  
October 2012.

[I-D.previdi-isis-te-metric-extensions]

Previdi, S., Giacalone, S., Ward, D., Drake, J., Atlas,  
A., and C. Filsfils, "IS-IS Traffic Engineering (TE)  
Metric Extensions",  
draft-previdi-isis-te-metric-extensions-02 (work in  
progress), October 2012.

[RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation  
Element (PCE)-Based Architecture", RFC 4655, August 2006.

[RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic  
Optimization (ALTO) Problem Statement", RFC 5693,  
October 2009.

#### Authors' Addresses

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

Email: shane@level3.net

Jan Medved  
Cisco Systems, Inc.  
170 West Tasman Drive  
San Jose, CA 95134  
USA

Email: jmedved@cisco.com

Stefano Previdi  
Cisco Systems, Inc.  
Via Del Serafico 200  
Rome 00144  
IT

Email: sprevidi@cisco.com

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

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

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 2, 2013

Fangwei. Hu  
ZTE  
Bhumip. Khasnabish  
ZTE USA Inc.  
Mar 2013

I2RS overlay use case  
draft-hu-i2rs-overlay-use-case-00.txt

Abstract

This document proposes an overlay network use case. The forwarding routers network is an overlay structure. There are two kinds of forwarding routers: Edge Router(ER) and Core Routers(CR). Edge Router encapsulates format data based on the tunnel type, which are established among Edge Routers. Core Router would be very simple and cheap. It focus on the encapsulation data forwarding. In order to reduce the equipment cost of Edge Routers, the network virtualization is provided in this document.

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 2, 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.

## Table of Contents

1. Introduction . . . . .	3
2. Overlay Network Structure . . . . .	3
2.1. Overview . . . . .	4
2.2. The Benefit of Overlay Network Structure . . . . .	5
2.3. Core Router Requirements . . . . .	5
2.4. Edge Router Requirement . . . . .	5
3. Network Virtualization(NV) . . . . .	6
3.1. Benefits of Network Virtualization . . . . .	6
3.2. Applications and Requirements . . . . .	6
3.3. Network Virtualization . . . . .	7
4. Security Considerations . . . . .	8
5. Acknowledgements . . . . .	8
6. IANA Considerations . . . . .	8
7. Normative References . . . . .	9
Authors' Addresses . . . . .	9

## 1. Introduction

As modern networks grow in scale and complexity, the need for rapid and dynamic control increases. I2RS([I2RS-FRM]) provide a new routing system framework to meet the requirement. There is a programmable interface for the forwarding router. All the forwarding routers should support I2RS agent to communicate with controllers. The forwarding routers gather the traffic and topology information, report to the controllers, and receive the policy from controllers.

Besides the idea of programmable and open interface, another key feature is forwarding plane and control plane separation in the I2RS and software define network. Some of the control and computing function could be separation from traditional routers. By this way, we hope that the service and data encapsulation are all done in the routers of the edge of network, and the routers in the core part are only focus on data forwarding. The core routers RIB table could only store the network(or equipment) IP prefix, and does not store user(or end station) IP prefix anymore. The RIB and FIB table capability of core routers would be reduced significantly, and the equipment cost could be lower. The full mesh tunnel is required for the edge Routers. This is actually an overlay network structure. The forwarding routers in the overlay network are divides into two kinds based on the roles in the network: CR(Core Router) and ER(Edge Router).The Edge Routers encapsulate the forwarding data based on the tunnel type, gather topology information, and report traffic to the controller, while Core Routers focus on fast data forwarding and receive only policy related information(metadata)from the controller.

## 2. Overlay Network Structure

## 2.1. Overview

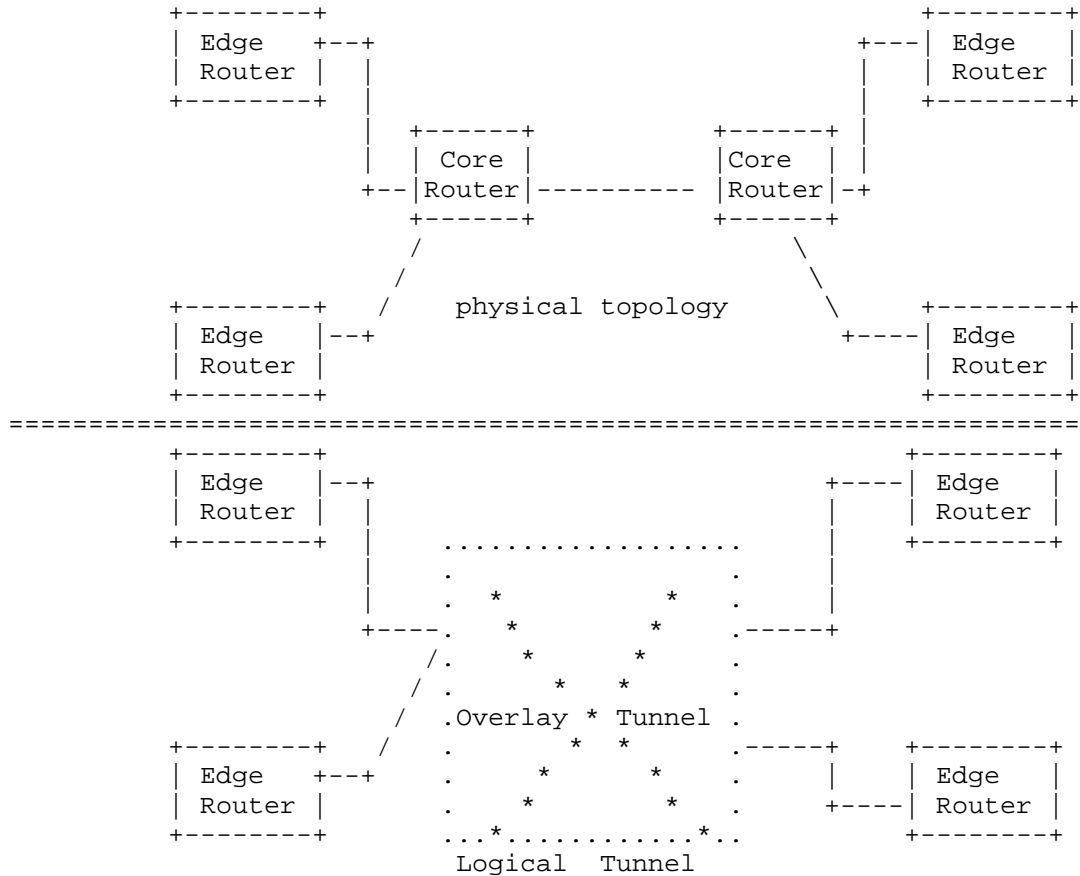


Figure 1 Overlay Structure

The overlay structure is shown in Figure 1. The upper half part of the Figure show a physical network. The Edge Routers are located in the edge of the overlay network, and are logically connected through Core Routers. The services and data encapsulation are done in the edge routers. The Core Routers are very simple; and focus on the data forwarding, and may not perceive any distinction among the tunnels to/from Edge Routers.

The lower half of the Figure shows a logical tunnel network. All the Edge Routers are connected via a logical-full mesh tunnel-based connection among them. The tunnel could be IP/MPLS/OTN tunnel. Edge Router encapsulates/decapsulates the data based on the tunnel type. If tunnel type is IP network, the encapsulation format would be IP

over GRE, or IP over UDP. If tunnel is MPLS network, the encapsulation format is IP over MPLS, which is similar to the MPLS data format. If tunnel is OTN, the tunnel format is IP/Ethernet over ODU.

## 2.2. The Benefit of Overlay Network Structure

- (1) Lower cost down for Core Routers: For the Core Router, it is not required to compute route, and distribute protocol signal. The Core Routers only store the equipment IP prefix, and do not store user IP prefix any more. The RIB and FIB table for core Router are very small. The size routing tables in the Core Routers does not increase and remains stable with the growth of the numbers of users.
- (2) Improved network security: The overlay network structure improves network security by splitting (and hence isolating) the provider equipment and user station. The attacks from hacker to core routers would therefore be separated by the edge routers.
- (3) Support of network virtualization: Some of the control and computing function could be separated from Edge Router and be done by controller. The edge router in the future is a hardware platform. The service, policy, and other control function, such as route computing, signal distribution can be furnished by special physical/virtual servers. The network virtualization for Edge Router is discussed in section 3.

## 2.3. Core Router Requirements

The Core Router performs the following functions:

- (1) Core Routers mainly focus on fast forwarding encapsulated data.
- (2) The control plane is very simple. It announces and floods the topology information.
- (3) For compatibility reasons, Route computation may be needed, but is not necessary.

## 2.4. Edge Router Requirement

The edge Router performs the following functions:

- (1) Access and resources management: Edge Routers support user Access authentication, authorization, and resource control and management. When there is new user access network, the edge router support user access authentication, authorization. If

the subscriber is legal and registered, he/she should should pass the access authentication and authorization tests.

- (2) Topology management: Edge Router gathers the network topology information and reports the topology to the controller. When the topology changes, the edge router reports the changes as well.
- (3) Policy management: Edge Router identifies the policy from The I2RS Commissioner([I2RS-Policy]).
- (4) Service management: Edge Router should identifies the services and performs the appropriate encapsulation.
- (5) Route and signal protocol management: Edge Router computes route based on the topology information received from other edge router and core router.
- (6) Tunnel control and management: Edge Routers manage and maintain tunnel information. All the edge routers are connected over logical full-mesh based tunnel network.
- (7) Traffic analysis and reporting: Edge router monitors the data traffic, and reports the traffic updates/changes.

### 3. Network Virtualization(NV)

#### 3.1. Benefits of Network Virtualization

- (1) NV reduces ER complexity and equipment costs.
- (2) NV allows flexibility and rapid deployment of new services; services can also be quickly scaled up/down based on demands.
- (3) Seamless support of scalability and reliability
- (4) NV allows flexibility and simplicity of function combination, for co-existence with hardware based network platform. An ER could be utilized both as BRAS, Firewall, or NAT equipment on the same hardware platform.

#### 3.2. Applications and Requirements

- (1) Tunnel gateway elements: IPSec/SSL VPN gateway.
- (2) Traffic analytics: DPI, QoS measurement, SLA agent.
- (3) Converged and network-wide functions: AAA Server, policy control and charging platform.
- (4) Security function: Firewalls, virus scanners, instruction detection and prevention systems.

### 3.3. Network Virtualization

Edge routers can support network virtualization, An ER can be a hardware based platform, and the other necessary functions can be supported via separate servers A programmable interface between functional server and edge router can be used to support this paradigm. When there is new service, we only need to add a new server to support that service, and there may be minimal or no changes required to the edge router.

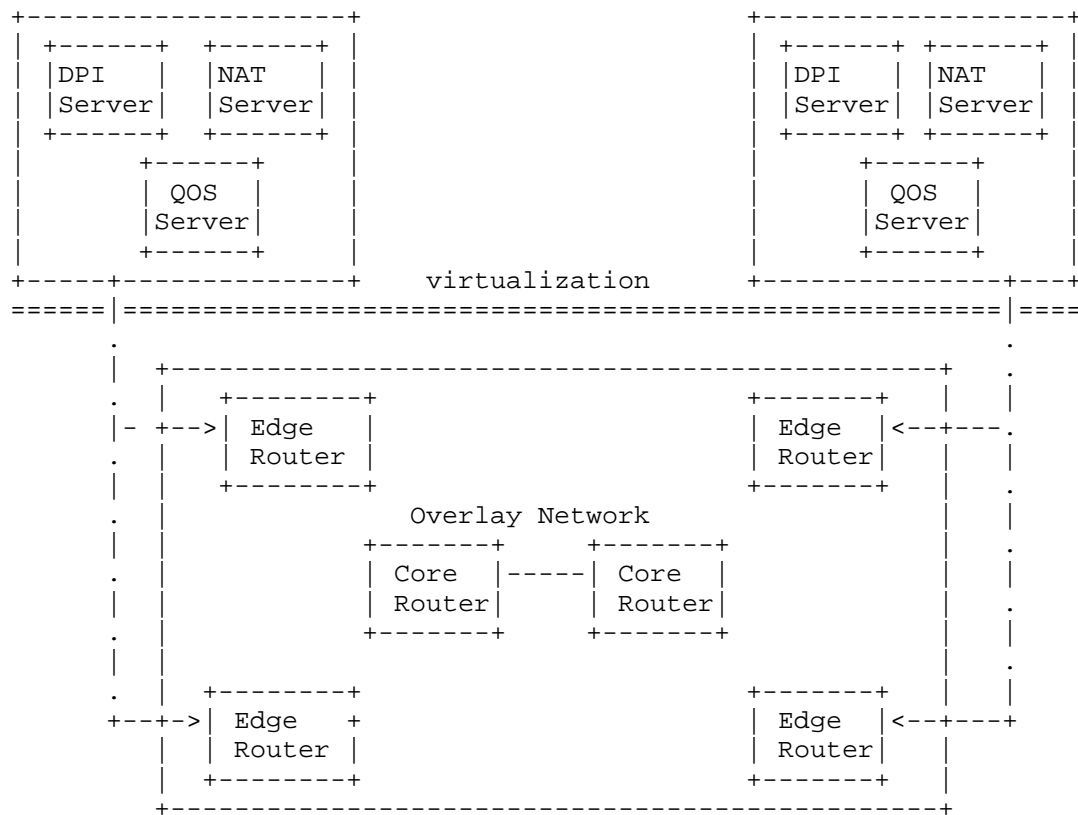


Figure 2: Network Virtualization

#### 4. Security Considerations

TBD

#### 5. Acknowledgements

TBD

#### 6. IANA Considerations

TBD

## 7. Normative References

### [I2RS-FRM]

Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Framework", draft-ward-i2rs-framework-01 (work in process), July 2012.

### [I2RS-Policy]

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 process), September 2012.

## Authors' Addresses

Fangwei Hu  
ZTE  
No.889 Bibo Rd  
Shanghai, 201203  
China  
  
Phone: +86 21 68896273  
Email: hu.fangwei@zte.com.cn

Bhumip Khasnabish  
ZTE USA Inc.  
55 Madison Avenue, Suite 160  
Morristown, New Jersey 07960  
USA  
  
Phone: +001-781-752-8003  
Email: Bhumip.khasnabish@zteusa.com



INTERNET-DRAFT  
Intended Status: Proposed Standard  
Expires: August 19, 2013

R. Fernando  
P. Chinnakannan  
M. Madhayyan  
A. Clemm  
February 15, 2013

YANG modifications for I2RS  
draft-rfernando-i2rs-yang-mods-00

Abstract

Interface to Routing Systems (I2RS) provides the mechanisms for the programmatic access of routing system components. YANG [RFC 6020] is a modeling language that is used to specify an external visible data model of sub-system - for defining both configuration and operational data held in the networking device. NETCONF is the protocol that is used to perform these configurations and accessing the operational data.

This document proposes to use the YANG data modeling language for I2RS data models. It also proposes a set of YANG language changes to enable the I2RS data models for multi-client and programmatic access.

This document also proposes the scope for the I2RS programmed data set on the Routing System and the necessary mechanisms for data synchronization.

This document also recommends a binary encoding for "on-the-wire" transfer of the YANG data model elements instead of XML.

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/lid-abstracts.html>

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

#### Copyright and License 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	Terminology . . . . .	4
2	I2RS Components . . . . .	4
2.1	Overview . . . . .	4
2.2	I2RS components . . . . .	4
3	I2RS extensions to Yang . . . . .	5
3.1	Overview . . . . .	5
3.2	Exclusive Owner I2RS . . . . .	7
3.3	Canonical Client Name Identifier (cname) . . . . .	11
3.4	Ephemeral data nodes . . . . .	14
3.5	YANG module "ietf-i2rs-extensions.yang" . . . . .	17
3.6	Client priority . . . . .	19
4	Security Considerations . . . . .	21
5	IANA Considerations . . . . .	21
6	References . . . . .	21
6.1	Normative References . . . . .	21
7	Authors' Addresses . . . . .	21

## 1 Introduction

The Network Configuration Protocol (NETCONF) provides mechanisms to programmatically install, manipulate, and delete the configuration of network devices. It uses an Extensible Markup Language (XML)-based data encoding for the configuration data as well as the protocol messages. The NETCONF protocol uses a remote procedure call (RPC) paradigm and protocol operations are performed on top of this simple RPC layer. A client encodes an RPC in XML and sends it to a server using a secure, connection-oriented session. The server responds with a reply encoded in XML.

YANG is the NETCONF Data Modeling Language [RFC6020], which supports hierarchical modeling of data elements that represent the configuration and runtime state of a particular network element.

The elements defined by the YANG module can be used for NETCONF-based communication, including passing configuration and state data, remote procedure calls (RPCs), and notifications. Thus YANG allows one to completely describe all the data sent between a NETCONF client and server.

YANG models the hierarchical organization of data as a tree in which each node has a name and a value or a set of child nodes. It provides clear and concise descriptions of the nodes, as well the inter-relationship between those nodes.

Interface to routing system (I2RS) framework is a draft proposal that sets out to build a framework to provide a common, standard interface to allow programmatic access to the information maintained in a router, like the routing protocol states, statistics, Routing Information Base (RIB) states, interface and their states etc.

Fundamental to the I2RS is a clear data model that defines the semantics of the information that can be written and read. This document proposes the use of Yang to define I2RS data models. It further proposes a set of YANG language extensions in order satisfy all the requirements of I2RS.

Specifically, this document makes the following proposal:

- o That I2RS shall use YANG for defining the data models that are applicable for I2RS. These models are termed 'I2RS data models' in this document.
- o A mechanism to express data ownership for a data set injected by I2RS clients into a Routing System using the I2RS data models.

- o A mechanism to express multi-client semantics and multi-client operations in an I2RS data model.
- o A mechanism to express the life time scope of programmatic data injected by I2RS clients in a Routing System and to express them in the I2RS data models.

The draft proposal for binary encoding for "on-the-wire" transfer of the YANG data model elements instead of XML and the associated NETCONF version2 is specified in a companion document (draft-tbd). The extensions proposed in this document is independent of the other changes proposed and will operate with or without the binary encoding as well as NETCONF version 2.

## 1.1 Terminology

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 I2RS Components

### 2.1 Overview

This section describes the roles performed by the different I2RS components when YANG is used as the modeling language and NETCONF used as the protocol.

### 2.2 I2RS components

An I2RS domain comprises of the following entities:

- o A Routing System supporting the I2RS data models and interfaces.
- o I2RS clients accessing and operating on the Routing systems using the I2RS interface.
- o I2RS servers present within a Routing system and providing the interface into the I2RS services and associated data models.
- o The NETCONF V2 Protocol that is used by the I2RS clients to access the I2RS services provided by the I2RS servers using a binary encoding format.
- o The I2RS client admission and authorization control service, which is co-located within the routing system. This function is provided by

an external service and this document recommends using NACM model [RFC 6536].

### 3 I2RS extensions to Yang

#### 3.1 Overview

I2RS clients operate on the I2RS data models to obtain the services provided by the Routing sub-systems. Client operations result in creation, modification and deletion of data sets in the I2RS data model tree, which is represented in YANG.

The data set injected by an I2RS client may be single valued, which is modeled as a "leaf" object or multi-valued, which are modeled as hierarchy of nodes using "containers", "lists", etc. The hierarchy of nodes, termed as "data model sub tree" or simply "sub tree", is conceptually owned by the I2RS client.

I2RS clients may access operational data and persistent configuration data. In this, they are like applications that automate management workflows. In addition, I2RS clients may inject ephemeral data. Ephemeral data corresponds to data that installs state, but that (unlike configuration data) is not persisted.

The ownership in this context requires identification of the I2RS client that injected the data set and the exclusivity assurance that the entire data set is injected and maintained by one client.

The exclusivity property guarantees that no other client shall modify any data item in the data set injected by one I2RS client. This concept of exclusivity assurance is proposed and described in section 3.2 Exclusive Owner, using the new YANG statement "exclusive-owner". That said, while an I2RS client is not able to modify the data that was injected by another client, it may be able to view data injected by other clients (assuming proper authorization). This is required so that I2RS clients are able to understand the I2RS server's behavior.

In contrast, current NETCONF treats all data to be global, meaning modification of data by one client is visible to all other clients and can be modified by other clients as well. In short, NETCONF is inherently designed for multi-owner datastores. Whether data items can be modified or not depends on the intrinsic nature of the data item (specified by 'config' statement). The proposed YANG extensions allow establishing a concept of ownership to data that is based on the extrinsic property of who created it, rather than solely based on the intrinsic nature of the data item. The I2RS data models may be operated upon by multiple I2RS clients or a NETCONF client or a

mixture of both types of clients.

I2RS also requires a uniform and consistent mechanism to identify the client that had injected the data set for the entire data model or to the subset of data model. This is required so that ownership of data items can be tagged to the data item themselves. The YANG data model language must provide a capability to express and capture the identity of a data model client in uniform and consistent manner across all data models. The data model client identifier must be a canonical client name identifier whose value must reference a unique client name record in a NETCONF access control model, NACM, [RFC 6536] database within the domain. Section, 3.4 Canonical Client Name Identifier (cname) describes this capability through the canonical client name, "cname" extension to YANG.

The I2RS clients operations on routing sub-systems results in creation, modification and deletion of data elements that are represented by the I2RS data models. The I2RS data models not only provide the schema for the data set injected, but also indicate the sections within the data model that are modifiable or editable by an I2RS client. The term editable is used in this document to refer to any IRS operation that leads to creation, modification or deletion of data model elements.

The editable portions of a data model sub tree is indicated by the YANG statement "config true", and any resulting data set through edit operations are stored in a data store called the running configuration data store. This data set is functionally permanent, in the sense that this data set must be stored in a persistent data store by the Routing System component and it is present across router reboots.

However, in I2RS operations, one of the requirement is to allow the i2rs clients to perform edit operations and the resulting data set is not stored permanently. This data set is "ephemeral" and is lost once a router reboots. This requirement is met using a new YANG extension called "ephemeral true", which is described in section 3.5 Ephemeral data nodes.

An I2RS client injects a set of data items under a sub tree within an I2RS data model. The exclusivity assurance guarantees that no other client will be able to modify any data item in the data set injected by one I2RS client in a sub tree that is indicated through the YANG statement "exclusive-owner".

It is possible for I2RS clients to attach data items marked as "ephemeral" or "exclusive" as a subtree underneath another data node, defined in an existing YANG data module. The fact that the I2RS

client injects data under an existing subtree MUST NOT affect the behavior of this containing subtree. Specifically, the exclusivity property MUST NOT prevent an item that was previously configurable, from being locked for configuration purposes. If a container is deleted, any I2RS client data attached below it is deleted along with it. In practice, it is therefore RECOMMENDED that I2RS ephemeral data is only attached below subtrees that are not subject to configuration (e.g., data items representing an immutable property, such as the system as a whole), or highly unlikely to change (e.g., data items representing a physical interface).

Although 'exclusivity' and 'ephemeral' statements describe two different characteristics, in practice they tend to apply to the same set of data nodes. That said, the concept of exclusivity is orthogonal to the concept of ephemeral data. This means that the "exclusive" property can be applied to data nodes marked as "config true" just as it can be applied to data nodes that are marked as ephemeral. Applying an exclusivity concept to configuration data implies the need to apply this concept across datastores (running, startup, candidate etc). As noted before, NETCONF datastores are essentially global and do not carry the notion of ownership to data.

### 3.2 Exclusive Owner I2RS

Currently NETCONF/YANG allows an instance of a sub tree, created by one client, to be modified later by another client. A data model primitive is required that disallows modifications to ephemeral data elements under a sub tree created by one I2RS client by another I2RS client.

This mechanism provides the guarantee that no other client shall modify any data item in the data set injected by one I2RS client.

This draft specification proposes that a new YANG statement be supported that defines a data node as "exclusive". The presence of the statement indicates that the data node, and all its descendants, is modifiable only by the client which created the instance. Other clients can retrieve the data node, but cannot edit it or delete it. They also cannot add any of their own data nodes below it.

There are several scenarios for applying exclusive ownership. The following outlines some of them:

- o Exclusive ownership shall apply for an entire list. This can be visualized as multiple identical tables, with each table being owned by one client. In this case, the YANG data model needs to be defined such that the list is contained in a container. The exclusive property is applied to that container.

o Exclusive ownership shall apply for a list element, but different list elements can have different owners. This can be visualized as a single table with multiple rows, with each row owned by a different client. In this case, the exclusive property is applied at the level of the definition of the list.

o Exclusive ownership shall apply only to a particular cell within a list element. This can be visualized as a single table with one cell of a row owned by a different clients. In this case, the exclusive property is applied at the level of the definition of the data node within the list.

The requirement for exclusive ownership is best illustrated with a few examples.

In the first example, the exclusive property is applied to a container within a list element. This means that within a list element, the container and everything it contains can only be modified by its owner. However, different list elements can have different owners for the corresponding container. Also, data nodes that are part of the same list element, but siblings of the container, can be modified by other clients as well.

```
module i2rs-routing
{
    imports ietf-i2rs-extensions {
        prefix "ix";
    }

    config true;

    description "An i2rs-routing module that contains sub tree nodes
that are either single-owner or multi-owner. This example is for
illustration of the single owner data node that represents a
sub-section of cells in a row";

    list i2rs-route {
        key "prefix";
        .....

        leaf prefix {
            type inet:ip-prefix;
        }

        container i2rs-route-attributes {
            ix:exclusive;
            ix:ephemeral;
        }
    }
}
```

```

    //This allows the name of the owning client
    //to be returned as part of the instantiated
    //information
    uses ix:client;

    description "The i2rs-route-attributes container contains
    several properties of a route. Only the client that
    originally created the container is allowed to modify or
    delete the container and its contents.";
}

container i2rs-dont-care {
    description "The i2rs-don't-care container contains
    properties of a route that can be manipulated
    by any client";
}
}
```

In the second example, the exclusive property is applied to list elements. Each list element has an exclusive owner. However, the same list can contain many list elements, and different list elements can have different exclusive owners.

```

module i2rs-routing
{
    imports ietf-i2rs-extensions {
        prefix "ix";
    }

    config true;

    description "An i2rs-routing module that contains sub tree
    nodes that are either single-owner or multi-owner. This
    example is for illustration of the single owner data node that
    represents a single row within a table";

    list i2rs-route {
        ix:exclusive;
        ix:ephemeral;
        key "prefix ix:cname";

        //This allows the name of the owning client
        //to be returned as part of the instantiated
        //information, and to be a part of the key
        uses ix:client;

        container i2rs-route-attributes {
```

```
description
"The i2rs-route-attributes container contains
several properties of a route. Each list element has
an exclusive owner. Only the client that created the
list element can edit or delete it. In case of
multiple list elements with the same prefix, the
element that is associated with the client of the
highest priority takes effect. ";
    }
}
```

Note that in the above example, the client name ("ix:cname") is used as a key. This allows two separate clients, distinguished by their cname, to inject a list element of otherwise the same key.

Note furthermore that in this example, if there are multiple list elements that have the same prefix as part of their index, only a single list element (i.e. only a single route entry) for the same prefix will be in effect. This is resolved by virtue of the owning client's priority, as determined by the server.

This example addresses the following scenario: Consider two I2RS clients, "sdn-app-A" and "sdn-app-b", operating on the above data model and injecting routes. Both clients may inject the same prefix "10.0.0.0/8" and will result into two separate data nodes in the hosting routing system component. The data record introduced by the I2RS client "sdn-app-A" is identified by the keys {10.0.0.0/8, "sdn-app-A"}, which contains an instance of the container i2rs-route-property and an instance of the container i2rs-don't-care. Likewise, the data record introduced by the I2RS client "sdn-app-B" is identified by the keys {10.0.0.0/8, "sdn-app-B"}.

It would have been possible to declare the same list without the client name as a key. In that case, each list element still has its own exclusive owner. However, two clients cannot own a list element that otherwise has the same key.

In the third example, the exclusive property is applied to the entire list. In this case, a container for the list is introduced and the exclusive property applied at that level.

```
module i2rs-routing
{
    imports ietf-i2rs-extensions {
        prefix "ix";
    }

    config true;

    description "An i2rs-routing module that contains sub tree
nodes that are either single-owner or multi-owner. This
example is for illustration of the single owner data node that
represents a conceptual table";

    container i2rs-routes {

        ix:exclusive;
        ix:ephemeral;
        uses ix:client;

        description
        "This container serves as the container of a list of
i2rs routes that has an exclusive owner. Only the
owner of this container can modify the list and any
list elements within it.";

        list i2rs-route {
            key "prefix";

            container i2rs-route-attributes {
                description "The i2rs-route-attributes container
contains several properties of a route. ";
            }
        }
    }
}
```

It should be noted that the concept of exclusive ownership pertains only with regards to modification operations, not to retrieval operations. Any client can retrieve data, even if it is exclusively owned by another client.

### 3.3 Canonical Client Name Identifier (cname)

A routing system supports data models for different components within the system which are known as I2RS data models in this specification. The I2RS data models may be operated by multiple I2RS clients or a single configuration NETCONF client or a mixture of both types of

clients. This programmatic access requires the concept of a client and associated properties to be specified in the data model. A data model requires a uniform and consistent methodology, and a mechanism to identify the client that had injected the data set for the entire data model or to the subset of the data model. The YANG data model language must provide a capability to express and capture the identity of a data model client in uniform and consistent manner across all data models.

This specification proposes the following elements to meet with the I2RS framework:

A new data type "cname" is introduced, used to identify the client that owns a particular data node.

The data model client identifier value must reference a unique client name record in a NETCONF access control model, [RFC 6536] database within the domain. The NACM service within the domain shall be used for specifying the client access privileges and other authentication, authorization records.

A grouping, "client" is introduced which contains a client identifier of cname type. The grouping may be used in conjunction with any data nodes (e.g. containers or lists) whose data owner needs to be identified. The client identifier is always provided by the server, reflecting the client name as identified through NACM.

The client identifier may be used as key for identifying the client in the data model sub tree sections that requires unambiguous identity of the data owner.

This document specifies that I2RS data models use "cname" type to specify the client identifier at required sections that are specified using the "exclusive" keyword.

There are times when data provided by multiple clients need to be maintained in a list with clear ownership of nodes maintained in the list. For instance, next-hops for the same prefix could be provided by multiple applications.

This type of data model construct can be achieved in two ways as follows:

- o Introducing an explicit YANG "list" node with a single key of the type "cname" as a parent node for a data model sub tree that is a non list type like containers, leaf-list, leaf etc.
- o Introducing an additional key of the type "cname" for a data model

sub tree that is a YANG list.

```
module i2rs-routing {  
  
    imports ietf-i2rs-extensions {  
        prefix "ix";  
    }  
  
    list i2rs-routing-client {  
        key "ix:cname";  
        ix:exclusive;  
  
        description "The i2rs-routing-client is a list node  
        that provides the client identifier of the I2RS client  
        performing programmatic operation below this  
        sub tree in the data model";  
  
        uses ix:client;  
  
        container i2rs-routing-table {  
  
            description "Routing table maintained per client";  
            /* Contents not specified here*/  
        }  
    }  
}
```

The following code fragment demonstrates a I2RS programmatic sub tree that is a list node.

```
module interfaces {  
  
    imports ietf-i2rs-extensions {  
        prefix "ix";  
    }  
  
    container a-non-i2rs-container {  
        /* Contents not specified */  
    }  
  
    list i2rs-routing-interface {  
        key "ix:cname if-name";  
        ix:exclusive;  
  
        description "i2rs-routing-interface provides a list of i2rs  
        interfaces. The I2RS client identifier performing  
        programmatic access to the routing interfaces are identified
```

```
    by the key client-id.";

    uses ix:client;

    leaf if-name {
        type string;

        description "The key field which is of the string type
            that specifies the interface name";
    }
    /* Other nodes not specified */
}
}
```

### 3.4 Ephemeral data nodes

The I2RS clients operations on routing sub-systems results into creation, modification and deletion of data elements that are represented by the I2RS data models. The I2RS data models not only provide the schema for the data set injected, but also indicate the sections within the data model that are modifiable or editable by an I2RS client.

The editable portions of a data model sub tree is indicated by the YANG statement "config true", and any resulting data set through edit operations are stored in a data store called the running configuration data store. This data set is functionally permanent, in the sense that this data set must be stored in a persistent data store by the Routing System component and it is present across router reboots. However, in Routing System operations a requirement is present that must allow clients to perform edit operations and the resulting data set is not stored permanently. This data set is ephemeral and is present only for the duration in which the Router System component is up and operational. This specification calls this persistency nature of the I2RS client state data on the routing system as ephemeral state and the data associated as ephemeral state data.

NETCONF allow its clients to create, modify, and delete data model elements expressed in YANG. This data created by NETCONF operations is called as configuration data and is associated with a certain type of data store called running configuration. The data present in the running configuration data store is permanent and is present across routing system reboots and is in contrast with the state data created and maintained by I2RS clients. This requires a new YANG extension to designate ephemeral state data. It must be noted that only sections of the data model that use the YANG statement "config true" are

editable. Other sections of the data model that use the YANG statement "config false" are not editable and these type of data are called as operational data. Operational data is not associated with any data store. I2RS clients need a third type of data which must be editable, but not persistent across routing system component reboots as well as not associated with any data store like the operational data.

This specification proposes a new YANG statement called "ephemeral" be supported which takes one argument that is a string with the value that "true" or "false". A value of "true" indicates the inclusive data node and all its sub tree is editable and remains persistent as long as the routing system component is up and operational. Value of 'false' serves the same purpose of 'config false' i.e. to mark a schema node as operational data.

The YANG statement "ephemeral true;" may be specified at any node of the data model tree and is inherited by all descendant nodes.

Ephemeral data cannot be part of a startup or candidate datastore. It can only be part of a running datastore.

The following example shows a valid use of the "ephemeral" statement.

```
module i2rs-routing {  
  
  description "An i2rs-routing module with the root node of  
    designated as configurable for illustration purpose";  
  
  container i2rs-routing-main-container {  
  
    config true;  
  
    container i2rs-section {  
      ix:ephemeral;  
  
      description "The node i2rs-section and its entire sub  
        tree is now made ephemeral. This indicates that this  
        node and all children of the container i2rs-section is  
        writable but not configurable any more. Any writes  
        made to this sub tree is not present in the running  
        configuration and is lost when the router reboots";  
    }  
  
    container config-section {  
      config true;  
    }  
  }  
}
```

```
        description "The config-section node and its sub tree
        is editable and is stored in the running config data
        store. This sub tree is stored persistently across
        reboots";
    }

    container oper-section {
        config false;

        description "This is node and its sub tree is not
        editable and is the operational data";
    }

    container an-ephemeral-node {
        ephemeral true;

        container incorrect-config-node {
            config true;

            description "This node is marked incorrectly
            as configurable and this is not permitted. A
            descendant node of an ephemeral schema node cannot
            be marked as 'config true'";
        }
    }
}
```

The following example shows how a section of the ephemeral sub tree can represent operational data.

```
module i2rs-routing {
    description "An i2rs-routing module with the root node of
    designated as configurable for illustration purpose";

    container i2rs-routing-main-container {
        config true;

        container another-ephemeral-node {
            ix:ephemeral;

            container an-intermediate-oper-node {
                config false;

                description "This node illustrates how an
                ephemeral sub tree can have operational data.";
            }
        }
    }
}
```

```

    }
  }
}

```

The following example shows in correct usage of the ephemeral statement that would be flagged as error by the YANG compiler.

```

module i2rs-routing {
  description "An i2rs-routing module with the root node of
    designated as configurable for illustration purpose";

  container i2rs-routing-main-container {
    config true;

    container an-ephemeral-node {
      ix:ephemeral;

      container incorrect-config-node {
        config true;

        description "This node is marked incorrectly
          as configurable and this is not permitted";
      }
    }

    container an-oper-node {
      config false;

      container an-errored-ephemeral-node {
        ix:ephemeral;

        description "This node is incorrect and will
          not be accepted by the data model development";
      }
    }
  }
}

```

### 3.5 YANG module "ietf-i2rs-extensions.yang"

Yang module "ietf-i2rs-extensions" contains a set of definitions needed by clients that intend to inject ephemeral state into a device. Specifically, this module defines the following:

- o A YANG extension that allows to declare the "ephemeral" clause
- o A YANG extension that allows to declare the "exclusive" clause

- o A grouping to be used in conjunction with data nodes that use the ephemeral clause. This grouping contains node to identify the owning control client and that client's priority.

- o A set of management information containing a registry of control clients, expiration timers, and operational status (TBD)

```
file "ietf-i2rs-extensions.yang"
```

```
module ietf-i2rs-extensions {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-i2rs-extensions";  
  
    prefix "i2rs-extn";  
  
    imports nacm {  
        prefix nacm;  
    }  
  
    organization  
        "example.com";  
  
    contact  
        "tbd";  
  
    description  
        "This module contains YANG extensions that would be needed to  
        model i2rs specific data models.";  
  
    revision "2013-02-14";  
  
    extension ephemeral {  
        description  
            "This extension can be used on schema nodes to indicate that  
            those data nodes and their descendant nodes do not survive  
            device reload."  
  
        argument access-specifier {  
            yin-element true;  
        };  
    }  
  
    extension exclusive {  
        description  
            "This extension can be used on schema nodes to indicate that  
            those data nodes and their descendant nodes can be created by  
            and written to by only one i2rs client. Other clients may have
```

```
        have read-only access to those data node.;
    }

    typedef cname {
        type leafref {
            path "/nacm:nacm/nacm:groups/nacm:group/nacm:name";
        }
        description
        "Control client ID by which the client is authenticated by the
        server.  An object of this type is always populated by the
        server, not by the invoking application.";
    }

    grouping client {
        leaf cname {
            config false;
            type cname;
        }
    }
}
```

### 3.6 Client priority

Multiple I2RS clients can inject ephemeral state in a device. With the exclusivity statement in the data model, the state injected by multiple clients can all be maintained concurrently on the device.

Applications in the device that are interested in ephemeral state injected by multiple clients can subscribe to receive notifications and read from the data store. In some cases, I2RS clients can inject conflicting information. For instance, we could have multiple I2RS clients programming different next- hop for the same destination prefix.

In some cases, the backend applications would need to prioritize one entry over the other. To do so, we may need some mechanism to assign a relative priority to each of the i2rs client.

The proposal is to assign a priority to each client in the NACM database. In I2RS datamodels that use the 'cname', backend applications can receive the priority setting along with the client-id.

To set the priority for a NACM user, the proposal is to enhance the NACM user-group entries with a priority setting. Here is the corresponding YANG snippet, to be incorporated into a corresponding

YANG module:

```
augment /nacm:nacm/nacm:groups/nacm:group {  
    leaf priority {  
        type uint32;  
  
        description  
        "Relative priority setting of a user-group (to which a i2rs  
        client gets assigned to".  
  
        default 0;  
    }  
}
```

Both the user-group and priority setting is passed to the backend application if the data model uses the 'exclusive true' keyword. It is entirely up to the backend application to act on this data.

#### 4 Security Considerations

This draft does not change the security characteristics of YANG.

#### 5 IANA Considerations

This draft does not have any IANA considerations.

#### 6 References

##### 6.1 Normative References

[IRS-FRMWK] A. Atlas, T. Nadeau, D. Ward, "Interface to the Routing System Framework", draft-ward-irs-framework-00

[IRS-FRMWK-REQ] R. Fernando et al, "IRS Framework Requirements", draft-rfernando-irs-framework-requirement-00

#### 7 Authors' Addresses

Rex Fernando  
Cisco Systems  
170 Tasman Dr  
San Jose  
EMail: rex@cisco.com

Palani Chinnakannan  
Cisco Systems  
170 Tasman Dr  
San Jose  
EMail: pals@cisco.com

Muthumayan Madhayyan  
Cisco Systems  
170 Tasman Dr  
San Jose  
EMail: muthu@cisco.com

Alexander Clemm  
Cisco Systems  
170 Tasman Dr  
San Jose

INTERNET DRAFT

draft-rfernando-i2rs-yang-mods-00

February 15, 2013

EMail: alex@cisco.com

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: August 27, 2013

A. Atlas, Ed.  
T. Nadeau  
Juniper Networks  
D. Ward  
Cisco Systems  
February 23, 2013

Interface to the Routing System Framework  
draft-ward-i2rs-framework-01

Abstract

This document describes a framework for a standard, programmatic interface for full-duplex state transfer in and out of the Internet's routing system. It provides some basic use-cases, lists the type of information that might be exchanged over the interface, and describes suggested functionality for the interface to the Internet routing system.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 27, 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.

## Table of Contents

1. Introduction . . . . .	3
1.1. Functional Overview . . . . .	3
1.2. Example Use-Cases . . . . .	4
2. Programmatic Interfaces . . . . .	5
3. Common Interface Considerations . . . . .	6
3.1. Capabilities . . . . .	6
3.2. Identity, Authorization, Authentication, and Security . . . . .	7
3.3. Speed and Frequency of State Installation . . . . .	8
3.4. Lifetime of I2RS-Installed Routing System State . . . . .	8
4. Bidirectional I2RS Services . . . . .	10
4.1. Static Routing . . . . .	11
4.1.1. Routing Information Base Service . . . . .	11
4.1.2. Label Forwarding Information Base Service . . . . .	12
4.1.3. Multicast Routing Information Base Service . . . . .	13
4.2. Beyond Destination-based Routing . . . . .	13
4.2.1. Policy-Based Routing Service . . . . .	13
4.2.2. QoS State . . . . .	13
4.3. Protocol Interactions . . . . .	14
4.3.1. IGP Services . . . . .	14
4.3.2. BGP Service . . . . .	15
4.3.3. PIM and mLDP Services . . . . .	15
4.4. Triggered Sessions and Signaling . . . . .	16
4.4.1. OAM-related Sessions Interface . . . . .	16
4.4.2. Dynamic Session Creation . . . . .	16
4.4.3. Triggered Signaling . . . . .	16
5. Services for Learned Information from the Routing System . . . . .	16
5.1. Efforts to Obtain Topological Data . . . . .	17
5.2. Measurements . . . . .	18
5.3. Events . . . . .	18
6. Manageability Considerations . . . . .	19
7. IANA Considerations . . . . .	19
8. Security Considerations . . . . .	19
9. Acknowledgements . . . . .	19
10. Informative References . . . . .	20
Authors' Addresses . . . . .	21

## 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 behaviour of the router.

This document sets out a framework for a common, standards-based interface to this information. This Interface to the Routing System (I2RS) facilitates control and diagnosis of the route 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 behaviour while leveraging the existing routing system.

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

### 1.1. Functional Overview

There are three key aspects to the I2RS. First, the interface is a programmatic interface meaning that it is 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.

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 network element's OS via the state-insertion aspects of the interface and that state could then be distributed in a routing or signaling protocol.

Where existing mechanisms can provide part of the desired functionality, the coverage and gaps are briefly discussed in this document.

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 is also 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 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. Example Use-Cases

A few brief examples of ways an application could use the I2RS are presented here. These are intended to give a sense of what could be done rather than to be primary and detailed motivational use-cases.

**Route Control via Indirection:** By enabling an application to install routes in the RIB, it is possible that when, for example, BGP resolves its IGP next-hop via the RIB, that could be to an application-installed route. In general, when a route is redistributed from one protocol to another, this is done via the RIB and such a route could have been installed via the I2RS interface.

**Policy-Based Routing of Unknown Traffic:** A static route, installed into the RIB, could direct otherwise unrecognized traffic towards an application, through whatever appropriate tunnel was required, for further handling. Such a static route could be programmed with indirection, so that its outgoing path is whatever is used by another particular route (e.g. to a particular server).

**Services with Fixed Hours:** If an application were to provide services only during fixed time-periods, the application could install both a specific route on the local router in the RIB and advertise the associated prefix as being attached to the local router via the IGP. If the application knew the fixed hours, the state so installed could be temporal and automatically removed at approximately the correct time.

**Traffic Mirroring:** The interface to the multicast RIB could be used to mirror a particular traffic flow to both its original destination and a data collector.

**Static Multicast Trees:** An application could set up static (or partially static) multicast flows via entries in the multicast RIB without requiring an associated multicast protocol. This could be useful in networks with a fixed topology and well-planned distribution tree that provides redundancy.

## 2. Programmatic Interfaces

A number of management interfaces exist today that allow for the indirect programming of the routing system. These include proprietary CLI, Netconf, and SNMP. However, none of these mechanisms allow for the direct programming of the routing system. Such asynchronous interfaces are needed to support dynamic time-based applications.

These interfaces should cater to how applications typically interact with other applications and network services rather than forcing them to use older mechanisms that are more complex to understand, implement, as well as operate. The interfaces should allow applications to have limited, filtered or abstracted knowledge of the network. Authorization and authentication are also critical so that the I2RS can be used by a network application that is not completely controlled by the network operator but is, nonetheless, given some access to I2RS.

One very critical component of the I2RS is developing standard data models with their associated semantics. While many routing protocols are standardized, associated data models for them are not yet

available. Instead, each router uses different information, mechanisms, and CLI which makes a standard interface for use by applications extremely cumbersome to develop and maintain. Well-known data modeling languages, such as YANG [RFC6020], exist, have some in-progress data models, and might be used for defining the necessary data models for I2RS; however, more investigation into alternatives is required. It is understood that some portion (hopefully a small subset) will remain as proprietary extensions; the data models must support future extensions and proprietary extensions.

Since the I2RS will need to support remote access between applications running on a host or server and routers in the network, at least one standard mechanism must be identified and defined to provide the transfer syntax, as defined by a protocol, used to communicate between the application and the routing system. Common functionality that I2RS needs to support includes acknowledgements, notifications, and request-reserve-commit.

Appropriate candidate protocols must be identified that reduce the effort required by applications and, preferably, are familiar to application developers. Ideally, this should not require that applications understand and implement existing routing protocols to interact with I2RS. These interfaces should instead be based on light-weight, rapidly deployable approaches; technology approaches must be evaluated but examples could include ReSTful web services, JSON, XMPP, and XML. These interfaces should possess self-describing attributes (e.g. a web services interface) so that applications can quickly query and learn about the active capabilities of a device.

It may be desirable to also define the local syntax (e.g. programming language APIs) that applications running local to a router can use.

Since evolution is anticipated in I2RS over time, it is important that versioning and backwards compatibility are basic supported functionality. Similarly, common consistent error-handling and acknowledgement mechanisms are required that do not severely limit the scalability and responsiveness of these interfaces.

### 3. Common Interface Considerations

#### 3.1. Capabilities

Capability negotiation is a critical requirement because different implementations and software versions will have different abilities. Similarly, applications may have different capabilities for receiving exported information.

An I2RS agent will have offer multiple services, each with their own set of capabilities. Such capabilities may include the particular data model and what operations can be performed at what scale.

The capabilities negotiated may be filtered based upon different information, such as the I2RS client application's authorization, I2RS client application's capabilities, and the desired granularity for abstraction which the I2RS client application understands. Different types of authorization may require the router to advertise different capabilities and restrictions.

The capability negotiation may take place at different levels of detail based upon the I2RS client and the specific functions in the I2RS that the I2RS client is negotiating. The network element and application must use the I2RS to agree upon the proper level of abstraction for the interaction. For example, when an application describes a route between two topological items, these items may vary in detail from a network domain's name at a high level, or down to the port forwarding specifics of a particular device.

The data-model and capabilities available for an element may depend upon whether the element is physical or virtual; the virtual/physical distinction does not matter to I2RS. Similarly, the location of the element may influence how an application converses with the associated network element.

### 3.2. Identity, Authorization, Authentication, and Security

The identity of applications that wish to manipulate or interrogate the state of the routing system must be appropriately authorized. Role-based authorization and authentication is necessary; however, there are different existing solutions to this that can be investigated for use in I2RS.

Being able to associate the state and the modifications to a state with a specific application would aid in troubleshooting and auditing of the routing system. By associating identity and authorization with installed state, other applications with appropriate authority can clean up state abandoned by failed I2RS client applications, if necessary.

Security of communication between the application and the router is also critical and must be considered in the design of the mechanisms to support these programmatic interfaces.

### 3.3. Speed and Frequency of State Installation

A programmatic interface does not by itself imply the frequency of state updates nor the speed at which the state installation is required. These are critical aspects of an interface and govern what an application can use the interface for. The difference between sub-second responsiveness to millions of updates and a day delay per update is, obviously, drastic. The key attributes of the programmatic interface are described in Section 5 of [I-D.atlas-i2rs-problem-statement] and include that the interface must be asynchronous.

For each service in I2RS, it will be necessary to specify expected scaling, responsiveness, and performance so that applications can understand the uses to which the I2RS can be used.

I2RS must support asynchronous real-time interactions between the I2RS client applications and I2RS agent on the network element. I2RS must assume that there are many unrelated applications that may be simultaneously using I2RS. This requirement for multi-headed control has a number of implications. First, the I2RS agent must do arbitration between state installed by different I2RS clients. Second, I2RS clients must be able to subscribe to change events that notify them about changes done to state by other I2RS clients, configuration, or dynamic routing.

Furthermore, I2RS should construct services that cater to different scaling and frequency of update parameters: e.g., slow, but detailed queries of the system, or fast yet higher level (less detailed) queries or modifications.

### 3.4. Lifetime of I2RS-Installed Routing System State

In routers today, the lifetime of different routing state depends upon how that state was learned and committed. If the state is configuration state, then it is ephemeral when just in the running configuration or persistent when written to the startup configuration. If the state is learned via a routing protocol or SNMP, it is ephemeral, lasting only until the router reboots or the state is withdrawn.

Unlike previous injection mechanisms that implied the state lifetime, I2RS requires that multiple models be supported for the lifetime of state it installs. This is because the lifetime or persistence of state of the routing system can vary based on the application that programmed it, policies or security authorization of the application.

To provide flexibility, pre-programming, and handle dependencies, it

is necessary to have multiple models of when a operation is to be handled. Similarly, there are multiple models for when an operation is to expire.

There are three aspects to be considered.

Persistence ? : Does state installed survive reboot?

Persistent: State installed by the I2RS client remains on the I2RS agent's network element across reboots or restarts of the system. The installed state can be dynamically removed or manipulated by an application, by configuration, or by the routing system itself. This state does not appear in the router's configuration; it is processed after all the configuration upon a reboot.

Ephemeral: State installed by the I2RS client remains on the I2RS agent's network element in its active memory until such time as the installed state is either removed by a routing or signaling protocol, removed by a configuration initiated by an application, or the router reboots. In the case of the latter, past state is forgotten when the router reboots.

Operation Start-Time: There are different models for when an I2RS agent should start an I2RS operation.

Immediate: When the operation is received, it should be acted upon as quickly as reasonable (e.g. queued with other outstanding requests if necessary).

Temporal: An application may provide an operation that is to be initiated at a particular time. When the specified time is reached, the operation should be acted upon as quickly as reasonable. Implementations may, of course, strive to improve the time-accuracy at which the operation is initiated.

Triggered: The operation should be initiated when the specified triggering event has happened. A triggering event could be the successful or failed completion of another operation. A triggering event could be a system event, such as an interface up or down, or another event such as a particular route changing its next-hops.

State Expiration: When state is installed by an I2RS client, there are two different models to consider for when that state is to be removed.

**Temporal:** When state is installed by an I2RS client, it has an expiration time specified. When that time has passed, the I2RS agent removes that state from the network element. The state can also be dynamically removed or manipulated by an I2RS client, by configuration or the routing system itself.

**Unbounded:** When state is installed by an I2RS client, that state does not explicitly expire. The state can be dynamically removed or manipulated by an I2RS client, by configuration, or by the routing system itself.

Because it is possible to request operations in models other than "Immediate" and some of the start-times will be at an unknown future point (e.g. "Triggered"), it is not feasible to guarantee that the resources required by an operation will always be available without reserving them from the time the operation is received. While that type of resource reservation should be possible, I2RS clients must also be able to handle an operation failing or being preempted due to resources or due to a higher priority or better authorized I2RS client taking ownership of the associated state or resource.

#### 4. Bidirectional I2RS Services

I2RS is a bidirectional programmatic interface that allows both routing and non-routing applications to install, remove, read, and otherwise manipulate the state of the routing system.

Just as the Internet routing system is not a single protocol or implementation layer, neither does it make sense for the I2RS to be at a single layer or reside within a single protocol. For each protocol or layer, there are different data models, abstractions and interface syntaxes and semantics required. However with this in mind, it is ideal that a minimal set of mechanism(s) to define, transfer and manipulate this state will be specified with as few optional characteristics as possible. This will foster better interoperability between different vendor implementations.

Since I2RS is focused on the routing system, the layers of interest start with the RIB and continue up through the IGPs, BGP, RSVP-TE, LDP, etc. The intent is neither to provide I2RS services to the forwarding plane nor to provide I2RS services to application layers.

It is critical that these I2RS services provide the ability to learn state, filtered by request, as well as to install state. I2RS assumes that there will be multiple applications using I2RS and therefore the ability to read state is necessary to fully know the network element's state. In general, if an I2RS service allows the

setting of state, the ability to read and modify that state is also necessary.

#### 4.1. Static Routing

The ability to specify static routes exists via CLI and MIBs but these mechanisms do not provide a programmatic interface. I2RS solves this problem by proposing interfaces to the RIB, LFIB, and Multicast RIBs.

By installing static routes into the RIB layer, I2RS is able to utilize the existing router OS and its mechanisms for distributing the selected routes into the FIB and LIB. This avoids the need to model or standardize the forwarding plane.

##### 4.1.1. Routing Information Base Service

The RIB is populated with routes and next-hops as supplied by configuration, management, or routing protocols. A route has a preference based upon the specific source from which the route was derived. Static routes, specified via CLI, can be installed with an appropriate preference. The FIB is populated by selecting from the RIB based on policy and tie-breaking criteria.

The I2RS service should allow dynamic reading and writing of routes into the RIB. There are several important attributes associated with doing so, as follows:

**Preference Value:** This allows decisions between conflicting routes, whether I2RS-installed or otherwise. I2RS-installed routes can each be installed with a different preference value.

**Route Table Context:** There can be different route table contexts in the RIB. Examples include multiple protocols (e.g. IPv4, IPv6), multiple topologies, different uses, and multiple networks (e.g. VRF tables for VPNs). Appropriate application-level abstractions are required to describe the desired route table context.

**Route or Traffic Identification** The specific IP prefix or even interface must be specified.

**Outgoing Path and Encapsulation:** It is necessary to specify the outgoing path and associated encapsulation. This may be done directly or indirectly. This is one of the more complex aspects with the following considerations.

**Primary Next-Hops:** To support multi-path forwarding, multiple primary next-hops can be specified and the traffic flows split among them.

**Indirection:** Instead of specifying particular primary next-hops, it is critical to be able to provide the ability for indirection, such as is used between BGP routes and IGP routes. Thus, the outgoing path might be specified via indirection to be the same as another route's.

**Encapsulation:** Associated with each primary next-hop can be details on the type of encapsulation for the packet. Such encapsulation could be MPLS, GRE, etc. as supported by the router.

**Protection:** For fast-reroute protection, each primary next-hop may have one or more alternate next-hops specified. Those are to be used when the primary next-hop fails.

**DSCP:** For QoS, the desired DSCP to be used for the outgoing traffic can be specified.

It is useful for an application to be able to read out the RIB state associated with particular traffic and be able to learn both the preferred route and its source as well as other candidates with lower preference.

Although there is no standardized model or specification of a RIB, it may be possible to build an interoperable bi-directional service without one.

#### 4.1.2. Label Forwarding Information Base Service

The LFIB has a similar role to the RIB for MPLS labeled packets. Each entry has slightly different information to accommodate MPLS forwarding and semantics. Although static MPLS can be used to configure specific state into the LFIB, there is no bidirectional programmatic interface to program, modify, or read the associated state.

Each entry in the LFIB requires a MPLS label context (e.g. platform, per-interface, or other context), incoming label, label operation, and next-hops with associated encapsulation, label operation, and so on. Via the I2RS LFIB service, an application could supply the information for an entry using either a pre-allocated MPLS label or a newly allocated MPLS label that is returned to the application.

#### 4.1.3. Multicast Routing Information Base Service

There is no bidirectional programmatic interface to add, modify, remove or read state from the multicast RIB. This I2RS service would add those capabilities.

Multicast forwarding state can be set up by a variety of protocols. As with the unicast RIB, an application may wish to install a new route for multicast. The state to add might be the full multicast route information - including the incoming interface, the particular multicast traffic (e.g. (source, group) or MPLS label), and the outgoing interfaces and associated encapsulations to replicate the traffic too.

The multicast state added need not match to well-known protocol installed state. For instance, traffic received on an specified set, or all, interfaces that is destined to a particular prefix from all sources or a particular prefix could be subject to the specified replication.

#### 4.2. Beyond Destination-based Routing

Routing decisions and traffic treatment is not merely expressable via destination-based routing or even (S, G) routing, such as in multicast. Capturing these aspects into appropriate interfaces for the I2RS provides the ability for applications to control them as well.

##### 4.2.1. Policy-Based Routing Service

A common feature of routers is the ability to specify policy-based routing (PBR) rules for accepting, dropping, or differently forwarding particular traffic. This is a very useful functionality for an application to be able to rapidly add and remove state into. Such state would indicate the particular traffic to be affected and its subsequent behavior (e.g. drop, accept, forward on specified outgoing path and encapsulation, QoS, DSCP marking, policing, etc.). Such state is made more complex by the potential importance of ordering among the PBR rules.

While PBR rules can be specified via CLI, this mechanism is not a streaming programmatic interface nor is there generally the ability to specify particular time-based lifetimes for each rule.

##### 4.2.2. QoS State

While per-hop behaviors are defined as well as standard DSCP meanings, the details of QoS configuration are not standardized and

can be highly variable depending upon platform. It is NOT a goal of this work to standardize QoS configurations. Instead, a data object model can define push/pull configurations. More investigation is needed to better describe the details.

#### 4.3. Protocol Interactions

Providing I2RS interfaces to the various routing protocols allows applications to specify policy, local topology changes, and availability to influence the routing protocols in a way that the detailed addition or modification of routes in the RIB does not.

The decision to distribute the routing state via a routing or signaling protocol depends upon the protocol-layer at which this state is injected into the routing system. It may also depend upon which routing domain or domains this information is injected as well.

In addition it is necessary to have the ability to pull state regarding various protocols from the router, a mechanism to register for asynchronous events, and the means to obtain those asynchronous events. An example of such state might be peer up/down.

##### 4.3.1. IGP Services

The lack of a programmatic interface to the IGPs limits the ability of applications to influence and modify the desired behavior of the IGP.

An application may need to indicate that a router is overloaded (via ISIS or the method described in [RFC3137]) because that router does not yet have sufficient state synchronized or installed into it. When critical state is provided not merely by routers but also from applications via the I2RS, a synchronization mechanism can be needed.

The ability for an application to modify the local topology can be part of this interface. One possibility is to allow modification of local interface metrics to generally influence selected routes. A more extensive interface might include the ability to create a OSPF or ISIS adjacency across a specified interface (virtual or real) with the appropriate associated encapsulation.

The ability to attach a prefix to the local router would provide a straightforward method for an application to program a single router and have the proper routes computed and installed by all other routers in the relevant domains. Additional aspects to the prefix attachment, such as the metric with which to attach the prefix and fast-reroute characteristics, would be part of the interface.

Beyond such pure routing information, the need for an application to be able to install state to be flooded via an IGP has already been recognized. [I-D.ietf-isis-genapp] specifies a mechanism for flooding generalized application information via ISIS, but does not describe how an application can generate or consume this information. Similarly, [RFC5250] specifies Opaque LSAs for OSPF to provide for application-specific information to be flooded. An I2RS service and associated data object model would provide such a mechanism.

Additional investigation will identify other state that applications may wish to install.

From the IGP, applications via I2RS can extract significant topological information about the routers, links, and associated attributes.

#### 4.3.2. BGP Service

BGP carries significant policy and per-application specific information as well as internet routes. A significant service to BGP is expected, with different data object models for different applications. For example, the I2RS service to BGP could provide the ability to specify the policy on which paths BGP chooses to advertise. Additionally, the ability to specify information with an application-specified AFI/SAFI could provide substantial flexibility and control.

An existing example of application information carried in BGP is BGP Flowspec [RFC5575] which can be used to provide traffic filtering and aid in handling denial-of-service attacks.

The ability to extract information from BGP is also quite critical. A useful example of this is the information available from BGP via [I-D.gredler-idr-ls-distribution], which allows link-state topology information to be carried in BGP.

#### 4.3.3. PIM and mLDP Services

For PIM and mLDP, there are at least two types of state that an application might wish to install. First, an application might add an interface to join a particular multicast group. Second, an application might provide an upstream route for traffic to be received from - rather than having PIM or mLDP need to consult the unicast RIB.

Additional investigation will identify other state that applications may wish to install.

#### 4.4. Triggered Sessions and Signaling

##### 4.4.1. OAM-related Sessions Interface

An application may need to trigger new OAM sessions (e.g. BFD, VCCP, etc.) using an appropriate template. For example, there may be applications that need to create a new tunnel, verify its functionality via new triggered OAM sessions, and then bring it into service if that OAM indicates successful functionality. More investigation is needed to better describe the details.

##### 4.4.2. Dynamic Session Creation

An application may wish to trigger a peering relationship for a protocol. For instance, a targeted LDP session may be required to exchange state installed locally with a remote router. More investigation is needed to better describe the different cases and details.

##### 4.4.3. Triggered Signaling

To easily create dynamic state throughout the network, an application may need to trigger signaling via protocols such as RSVP-TE. An example of such an application can be a Stateful Path Computation Element (PCE)[I-D.ietf-pce-stateful-pce], which has control of various LSPs that need to be signaled.

More investigation is needed to better describe the different cases and details.

#### 5. Services for Learned Information from the Routing System

Just as applications need to inject state into the routing system to meet various application-specific and policy-based requirements, it is critical that applications be able to also extract necessary state from the routing system.

A part of each of these services is the ability to specify the generation of the desired information (e.g., collecting specific per-flow measurements) and the ability to specify appropriate filters to indicate the specifics and abstraction level of the information to be provided

The types of information to extract can be generally grouped into the following different categories.

**Topological:** The need to understand the network topology, at a suitable abstraction layer, is critical to applications. Connectivity is not sufficient - the associated costs, bandwidths, latencies, etc. are all important aspects of the network topology that strongly influence the decision-making and behavior of applications.

**Measurements:** Applications require measurements of traffic and network behavior in order to have a more meaningful feedback control loop. Such information may be per-interface, per-flow, per-firewall rule, per-queue, etc.

**Events:** There are a variety of asynchronous events that an application may require or use as triggering conditions for starting other operations. An obvious example is interface state events.

**Configuration:** For some aspects, it may be necessary for applications to be able to learn about the routing configuration on a box. This is partially available via various MIBs and NetConf. What additional information needs to be exported and the appropriate mechanisms needs further examination.

The need to extract information from the network is not new; there is on-going work in the IETF in this area. This framework describes those efforts in the context of the above categories and starts the discussion of the aspects still required.

### 5.1. Efforts to Obtain Topological Data

Topological data can be defined and presented at different layers (e.g. Layer-2, Layer-3) and with different characteristics exposed or hidden (e.g. physical or virtual, SRLGs, bandwidth, latency, etc.). It can also have different states, such as configured but unavailable, configurable, active, broken, administratively disabled, etc.

To solve the problem of only being able to obtain topological data via listening to the IGP in each area, BGP-LS [I-D.gredler-idr-ls-distribution] defines extensions to BGP so that link-state topology information can be carried in BGP and a single BGP listener in the AS can therefore learn and distribute the entire AS's current link-state topology. BGP-LS solves the problem of distributing topological information throughout the network. While I2RS may expand the information to be distributed, I2RS addresses the API aspect of BGP-LS and not the network-wide distribution.

At another level, ALTO [RFC5693] provides topological information at

a higher abstraction layer, which can be based upon network policy, and with application-relevant services located in it. The mechanism for ALTO obtaining the topology can vary and policy can apply to what is provided or abstracted.

Neither of these fully meet the need to obtain detailed, layered topological state that provides more information than the current functional status. While there are currently no sufficiently complete standards, the need for such functionality can be deduced by the number of proprietary systems that have been developed to obtain and manage topology; even Element Management Systems start with the need for learning and manipulating the topology. Similarly, orchestration layers for applications start with the need to manage topology and the associated database.

Detailed topology includes aspects such as physical nodes, physical links, virtual links, port to interface mapping, etc. The details should include the operational and administrative state as well as relevant parameters ranging from link bandwidth to SRLG membership. Layering is critical to provide the topology at the level of abstraction where it can be easily used by the application.

A key aspect of this service is the ability to easily rate-limit, filter and specify the desired information to be extracted. This will help in allowing the service to scale when queries are done.

## 5.2. Measurements

IPFIX [RFC5470] provides a way to measure and export per-traffic flow statistics. Applications that need to collect information about particular flows thus have a clear need to be able to install state to configure IPFIX to measure and export the relevant flows to the appropriate collectors.

## 5.3. Events

A programmatic interface for application to subscribe to asynchronous events is necessary. In addition to the interface state events already mentioned, an application may wish to subscribe to certain OAM-triggered events that aren't otherwise exported.

A RIB-based event could be reporting when the next-hops associated with a route have changed. Other events could be used to verify that forwarding state has been programmed. For example, an application could request an event whenever a particular route in the RIB has its forwarding plane installation completed.

When an application registers for events, the application may request

to get only the first such event, all such events, or all events until a certain time.

The full set of such events, that are not specifically related to other services, needs to be investigated and defined.

## 6. Manageability Considerations

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

**Data Authorization Levels:** The data-models used for I2RS need the ability to indicate the required authorization level for installing or reading a particular subset of data. This allows control of what interactions each application can have.

**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 critical.

**Configuration Interactions:** The interaction of state installed via the I2RS and via a router's configuration needs to be clearly defined.

## 7. IANA Considerations

This document includes no request to IANA.

## 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 3.2. Security of communications from the applications is also required.

More specifics on the security requirements requires further investigation.

## 9. Acknowledgements

The authors would like to thank Ken Gray, Adrian Farrel, Bruno Rijsman, Rex Fernando, Jan Medved, John Scudder, and Hannes Gredler

for their suggestions and review.

## 10. 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), February 2013.
- [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.
- [I-D.ietf-isis-genapp]  
Ginsberg, L., Previdi, S., and M. Shand, "Advertising Generic Information in IS-IS", draft-ietf-isis-genapp-04 (work in progress), November 2010.
- [I-D.ietf-pce-stateful-pce]  
Crabbe, E., Medved, J., Minei, I., and R. Varga, "PCEP Extensions for Stateful PCE", draft-ietf-pce-stateful-pce-02 (work in progress), October 2012.
- [RFC3137] Retana, A., Nguyen, L., White, R., Zinin, A., and D. McPherson, "OSPF Stub Router Advertisement", RFC 3137, June 2001.
- [RFC5250] Berger, L., Bryskin, I., Zinin, A., and R. Coltun, "The OSPF Opaque LSA Option", RFC 5250, July 2008.
- [RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", RFC 5470, March 2009.
- [RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, August 2009.
- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the

Network Configuration Protocol (NETCONF)", RFC 6020,  
October 2010.

Authors' Addresses

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

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

Thomas 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](mailto:wardd@cisco.com)

