

I2RS  
Internet-Draft  
Intended status: Informational  
Expires: March 1, 2015

J. Clarke  
G. Salgueiro  
C. Pignataro  
Cisco  
August 28, 2014

Interface to the Routing System (I2RS) Traceability: Framework and  
Information Model  
draft-clarke-i2rs-traceability-03

Abstract

This document describes a framework for traceability in the Interface to the Routing System (I2RS) and information model for that framework. It specifies the motivation, requirements, use cases, and defines an information model for recording interactions between elements implementing the I2RS protocol. This framework provides a consistent tracing interface for components implementing the I2RS architecture to record what was done, by which component, and when. It aims to improve the management of I2RS implementations, and can be used for troubleshooting, auditing, forensics, and accounting purposes.

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 March 1, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

## Table of Contents

1. Introduction . . . . .	2
2. Terminology and Conventions . . . . .	3
3. Motivation . . . . .	3
4. Use Cases . . . . .	3
5. Information Model . . . . .	4
5.1. I2RS Traceability Framework . . . . .	4
5.2. I2RS Trace Log Mandatory Fields . . . . .	5
5.3. End of Message Marker . . . . .	7
5.4. I2RS Trace Log Extensibility and Optional Fields . . . . .	7
6. Examples . . . . .	7
7. Operational Guidance . . . . .	7
7.1. Trace Log Creation . . . . .	8
7.2. Trace Log Temporary Storage . . . . .	8
7.3. Trace Log Rotation . . . . .	8
7.4. Trace Log Retrieval . . . . .	8
7.4.1. Retrieval Via Syslog . . . . .	9
7.4.2. Retrieval Via I2RS Information Collection . . . . .	9
7.4.3. Retrieval Via I2RS Pub-Sub . . . . .	9
8. IANA Considerations . . . . .	10
9. Security Considerations . . . . .	10
10. Acknowledgments . . . . .	10
11. References . . . . .	10
11.1. Normative References . . . . .	10
11.2. Informative References . . . . .	11
Authors' Addresses . . . . .	11

## 1. Introduction

The architecture for the Interface to the Routing System ([I-D.ietf-i2rs-architecture]) specifies that I2RS Clients wishing to retrieve or change routing state on a routing element **MUST** authenticate to an I2RS Agent. The I2RS Client will have a unique identity it provides for authentication, and should provide another, opaque identifier for applications (or actors) communicating through it. The programming of routing state will produce a return code containing the results of the specified operation and associated reason(s) for the result. All of this is critical information to be used for understanding the history of I2RS interactions.

This document describes use cases for I2RS traceability. Based on these use cases, the document proposes an information model and reporting requirements to provide for effective recording of I2RS interactions. In this context, effective troubleshooting means being able to identify what operation was performed by a specific I2RS Client, what was the result of the operation, and when that operation was performed.

Discussions about the retention of the data logged as part of I2RS traceability, while important, are outside of the scope of this document.

## 2. Terminology and Conventions

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

The architecture specification for I2RS [I-D.ietf-i2rs-architecture] defines additional terms used in this document that are specific to the I2RS domain, such as "I2RS Agent", "I2RS Client", etc. The reader is expected to be familiar with the terminology and concepts defined in [I-D.ietf-i2rs-architecture].

The IP addresses used in the example in this document correspond to the documentation address blocks 192.0.2.0/24 (TEST-NET-1), 198.51.100.0/24 (TEST-NET-2) and 203.0.113.0/24 (TEST-NET-3) as described in [RFC5737].

## 3. Motivation

As networks scale and policy becomes an increasingly important part of the control plane that creates and maintains the forwarding state, operational complexity increases as well. I2RS offers more granular and coherent control over policy and control plane state, but it also removes or reduces the locality of the policy that has been applied to the control plane at any individual forwarding device. The ability to automate and abstract even complex policy-based controls highlights the need for an equally scalable traceability function to provide event-level granularity of the routing system compliant with the requirements of I2RS (Section 5 of [I-D.ietf-i2rs-problem-statement]).

## 4. Use Cases

An obvious motivation for I2RS traceability is the need to troubleshoot and identify root-causes of problems in these increasingly complex routing systems. For example, since I2RS is a

high-throughput multi-channel, full duplex and highly responsive interface, I2RS Clients may be performing a large number of operations on I2RS Agents concurrently or at nearly the same time and quite possibly in very rapid succession. As these many changes are made, the network reacts accordingly. These changes might lead to a race condition, performance issues, data loss, or disruption of services. In order to isolate the root cause of these issues it is critical that a network operator or administrator has visibility into what changes were made via I2RS at a specific time.

Some network environments have strong auditing requirements for configuration and runtime changes. Other environments have policies that require saving logging information for operational or regulatory compliance considerations. These requirements therefore demand that I2RS provides an account of changes made to network element routing systems.

As I2RS becomes increasingly pervasive in routing environments, a traceability model offers significant advantages and facilitates the following use cases:

- o Automated event correlation, trend analysis, and anomaly detection.
- o Trace log storage for offline (manual or tools) analysis.
- o Improved accounting of routing system transactions.
- o Standardized structured data format for writing common tools.
- o Common reference for automated testing and incident reporting.
- o Real-time monitoring and troubleshooting.
- o Enhanced network audit, management and forensic analysis capabilities.

## 5. Information Model

### 5.1. I2RS Traceability Framework

This section describes a framework for I2RS traceability based on the I2RS Architecture. Some notable elements on the architecture are highlighted herein.

The interaction between the optional northbound actor, I2RS Client, I2RS Agent, the Routing System and the data captured in the I2RS trace log is shown in Figure 1.

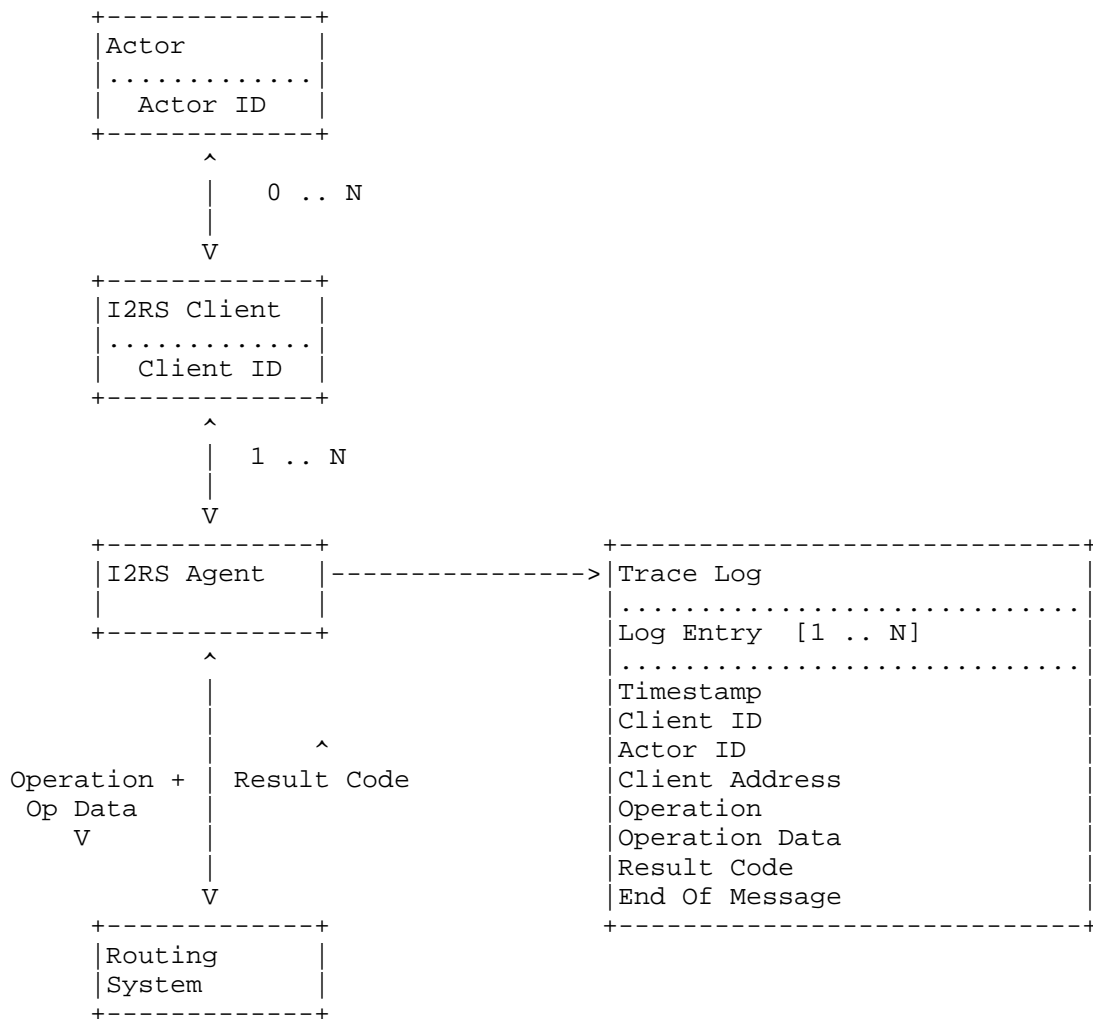


Figure 1: I2RS Interaction Trace Log Capture

## 5.2. I2RS Trace Log Mandatory Fields

In order to ensure that each I2RS interaction can be properly traced back to the Client that made the request at a specific point in time, the following information **MUST** be collected and stored by the Agent.

The list below describes the fields captured in the I2RS trace log.

**Entry ID:** This is a unique identifier for each entry in the I2RS trace log. Since multiple operations can occur from the same

client at the same time, it is important to have an identifier that can be unambiguously associated to a specific entry.

**Timestamp:** The specific time, adhering to [RFC3339] format, at which the I2RS transaction occurred. Given that many I2RS transactions can occur in rapid succession, the use of fractional seconds **MUST** be used to provide adequate granularity.

**Client Identifier:** The I2RS Client identifier used to authenticate the Client to the I2RS Agent.

**Actor Identifier:** This is an opaque identifier that may be known to the Client from a northbound controlling application. This is used to trace the northbound actor driving the actions of the Client. The Client may not provide this identifier to the Agent if there is no external actor driving the Client. However, this field **MUST** be logged. If the Client does not provide an actor ID, then the Agent **MUST** log an UNAVAILABLE value in the field.

**Client Address:** This is the network address of the client that connected to the Agent. For example, this may be an IPv4 or IPv6 address. [Note: will I2RS support interactions that have no network address? If so this field will need to be updated.]

**Operation:** This is the I2RS operation performed. For example, this may be an add route operation if a route is being inserted into a routing table.

**Operation Data:** This field comprises the data passed to the Agent to complete the desired operation. For example, if the operation is a route add operation, the Operation Data would include the route prefix, prefix length, and next hop information to be inserted as well as the specific routing table to which the route will be added. The operation data can also include interface information. Some operations may not provide operation data, and in those cases this field **MUST** be logged as a NULL string.

**Result Code:** This field holds the result of the operation. In the case of RIB operations, this **MUST** be the return code as specified in Section 4 of [I-D.nitinb-i2rs-rib-info-model]. The operation may not complete with a result code in the case of a timeout. If the operation fails to complete, it **MUST** still log the attempted operation with an appropriate result code (e.g., a result code indicating a timeout).

**End Of Message:** Each log entry **SHOULD** have an appropriate End Of Message (EOM) indicator. See section Section 5.3 below for more details.

### 5.3. End of Message Marker

Because of variability within I2RS trace log fields, implementors MUST use a format-appropriate end of message (EOM) indicator in order to signify the end of a particular record. That is, regardless of format, the I2RS trace log MUST provide a distinct way of distinguishing between the end of one record and the beginning of another. For example, in a linear formatted log (similar to syslog) the EOM marker may be a newline character. In an XML formatted log, the schema would provide for element tags that denote beginning and end of records. In a JSON formatted log, the syntax would provide record separation (likely by comma-separated array elements).

### 5.4. I2RS Trace Log Extensibility and Optional Fields

[NOTE: This section is TBD based on further development of I2RS WG milestones.]

## 6. Examples

Here is a proposed sample of what the fields might look like in an I2RS trace log. This is only an early proposal. These values are subject to change.

```
Entry ID:      1
Timestamp:     2013-09-03T12:00:01.21+00:00
Client ID:     5CEF1870-0326-11E2-A21F-0800200C9A66
Actor ID:      com.example.RoutingApp
Client Address: 192.0.2.2
Operation:     ROUTE_ADD
Operation Data: PREFIX 203.0.113.0 PREFIX-LEN 24 NEXT-HOP
                198.51.100.1
Result Code:   SUCCESS(0)
```

## 7. Operational Guidance

Specific operational procedures regarding temporary log storage, rollover, retrieval, and access of I2RS trace logs is out of scope for this document. Organizations employing I2RS trace logging are responsible for establishing proper operational procedures that are appropriately suited to their specific requirements and operating environment. In this section we only provide fundamental and generalized operational guidelines that are implementation-independent.

### 7.1. Trace Log Creation

The I2RS Agent interacts with the Routing and Signaling functions of the Routing Element. Since the I2RS Agent is responsible for actually making the routing changes on the associated network device, it creates and maintains a log of transactions that can be retrieved to troubleshoot I2RS-related impact to the network.

### 7.2. Trace Log Temporary Storage

The trace information may be temporarily stored either in an in-memory buffer or as a file local to the Agent. Care should be given to the number of I2RS transactions expected on a given agent so that the appropriate storage medium is used and to maximize the effectiveness of the log while not impacting the performance and health of the Agent. Section 7.3 talks about rotating the trace log in order to preserve the transaction history without exhausting Agent or network device resources. It is perfectly acceptable, therefore, to use both an in-memory buffer for recent transactions while rotating or archiving older transactions to a local file.

It is outside the scope of this document to specify the implementation details (i.e., size, throughput, data protection, privacy, etc.) for the physical storage of the I2RS log file. Data retention policies of the I2RS traceability log is also outside the scope of this document.

### 7.3. Trace Log Rotation

In order to prevent the exhaustion of resources on the I2RS Agent or its associated network device, it is RECOMMENDED that the I2RS Agent implements trace log rotation. The details on how this is achieved are left to the implementation and outside the scope of this document. However, it should be possible to do file rotation based on either time or size of the current trace log. If file rollover is supported, multiple archived log files should be supported in order to maximize the troubleshooting and accounting benefits of the trace log.

### 7.4. Trace Log Retrieval

Implementors are free to provide their own, proprietary interfaces and develop custom tools to retrieve and display the I2RS trace log. These may include the display of the I2RS trace log as Command Line Interface (CLI) output. However, a key intention of defining this information model is to establish an implementor-agnostic and consistent interface to collect I2RS trace data. Correspondingly, retrieval of the data should also be made implementor-agnostic.



The following three sections describe potential ways the trace log can be accessed. At least one of these three MUST be used, with the I2RS mechanisms being preferred as they are implementor-independent approaches to retrieving the data.

#### 7.4.1. Retrieval Via Syslog

The syslog protocol [RFC5424] is a standard way of sending event notification messages from a host to a collector. However, the protocol does not define any standard format for storing the messages, and thus implementors of I2RS tracing would be left to define their own format. So, while the data contained within the syslog message would adhere to this information model, and may be consumable by a human operator, it would not be easily parseable by a machine. Therefore, syslog MAY be employed as a means of retrieving or disseminating the I2RS trace log contents.

#### 7.4.2. Retrieval Via I2RS Information Collection

Section 6.7 of the I2RS architecture [I-D.ietf-i2rs-architecture] defines a mechanism for information collection. The information collected includes obtaining a snapshot of a large amount of data from the network element. It is the intent of I2RS to make this data available in an implementor-agnostic fashion. Therefore, the I2RS trace log SHOULD be made available via the I2RS information collection mechanism either as a single snapshot or via a subscription stream.

#### 7.4.3. Retrieval Via I2RS Pub-Sub

Section 6.7 of the I2RS architecture [I-D.ietf-i2rs-architecture] goes on to define a publish-subscribe mechanism for a feed of changes happening within the I2RS layer. I2RS Agents SHOULD support publishing I2RS trace log information to that feed as described in that document. Subscribers would then receive a live stream of I2RS interactions in trace log format and could flexibly choose to do a number of things with the log messages. For example, the subscribers could log the messages to a datastore, aggregate and summarize interactions from a single client, etc. Using pub-sub for the purpose of logging I2RS interactions augments the areas described by [I-D.camwinget-i2rs-pubsub-sec]. The full range of potential activities is virtually limitless and the details of how they are performed are outside the scope of this document, however.

## 8. IANA Considerations

This document makes no request of IANA.

## 9. Security Considerations

The I2RS trace log, like any log file, reveals the state of the entity producing it as well as the identifying information elements and detailed interactions of the system containing it. The information model described in this document does not itself introduce any security issues, but it does define the set of attributes that make up an I2RS log file. These attributes may contain sensitive information and thus should adhere to the security, privacy and permission policies of the organization making use of the I2RS log file.

It is outside the scope of this document to specify how to protect the stored log file, but it is expected that adequate precautions and security best practices such as disk encryption, appropriately restrictive file/directory permissions, suitable hardening and physical security of logging entities, mutual authentication, transport encryption, channel confidentiality, and channel integrity if transferring log files. Additionally, the potentially sensitive information contained in a log file SHOULD be adequately anonymized or obfuscated by operators to ensure its privacy.

## 10. Acknowledgments

The authors would like to thank Alia Atlas for her initial feedback and overall support for this work. Additionally, the authors acknowledge Alvaro Retana, Russ White, Matt Birkner, Jeff Haas, Joel Halpern and Dean Bogdanovich for their reviews, contributed text, and suggested improvements to this document.

## 11. References

### 11.1. Normative References

[I-D.ietf-i2rs-architecture]

Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-05 (work in progress), July 2014.

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

Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Problem Statement", draft-ietf-i2rs-problem-statement-04 (work in progress), June 2014.

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

## 11.2. Informative References

- [I-D.camwinget-i2rs-pubsub-sec]  
Beck, K., Cam-Winget, N., and D. McGrew, "Using the Publish-Subscribe Model in the Interface to the Routing System", draft-camwinget-i2rs-pubsub-sec-00 (work in progress), July 2013.
- [I-D.nitinb-i2rs-rib-info-model]  
Bahadur, N., Folkes, R., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-nitinb-i2rs-rib-info-model-02 (work in progress), August 2013.
- [RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.
- [RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, March 2009.
- [RFC5737] Arkko, J., Cotton, M., and L. Vegoda, "IPv4 Address Blocks Reserved for Documentation", RFC 5737, January 2010.

## Authors' Addresses

Joe Clarke  
Cisco Systems, Inc.  
7200-12 Kit Creek Road  
Research Triangle Park, NC 27709  
US

Phone: +1-919-392-2867  
Email: jclarke@cisco.com

Gonzalo Salgueiro  
Cisco Systems, Inc.  
7200-12 Kit Creek Road  
Research Triangle Park, NC 27709  
US

Email: gsalguei@cisco.com

Carlos Pignataro  
Cisco Systems, Inc.  
7200-12 Kit Creek Road  
Research Triangle Park, NC 27709  
US

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

Network Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: September 10, 2015

A. Clemm  
J. Medved  
R. Varga  
T. Tkacik  
Cisco  
N. Bahadur  
Bracket Computing  
H. Ananthakrishnan  
Packet Design  
March 9, 2015

A Data Model for Network Topologies  
draft-clemm-i2rs-yang-network-topo-04.txt

## Abstract

This document defines an abstract (generic) YANG data model for network/service topologies and inventories. The model serves as a base model which is augmented with technology-specific details in other, more specific topology and inventory models.

## 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 10, 2015.

## Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

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

#### Table of Contents

1. Introduction . . . . .	3
2. Definitions and Acronyms . . . . .	7
3. Model Structure Details . . . . .	7
3.1. Base Network Model . . . . .	7
3.2. Base Network Topology Model . . . . .	9
3.3. Extending the model . . . . .	11
3.4. Discussion and selected design decisions . . . . .	11
3.4.1. Container structure . . . . .	12
3.4.2. Underlay hierarchies and mappings . . . . .	12
3.4.3. Use of groupings . . . . .	12
3.4.4. Cardinality and directionality of links . . . . .	13
3.4.5. Multihoming and link aggregation . . . . .	13
3.4.6. Mapping redundancy . . . . .	13
3.4.7. Typing . . . . .	14
3.4.8. Representing the same device in multiple networks . . . . .	14
3.5. Items for further discussion . . . . .	15
4. YANG Modules . . . . .	16
4.1. Defining the Abstract Network: network.yang . . . . .	16
4.2. Creating Abstract Network Topology: network-topology.yang . . . . .	18
5. Security Considerations . . . . .	23
6. Contributors . . . . .	23
7. Acknowledgements . . . . .	23
8. References . . . . .	23
8.1. Normative References . . . . .	23
8.2. Informative References . . . . .	24
Authors' Addresses . . . . .	24

## 1. Introduction

This document introduces an abstract (base) YANG [RFC6020] [RFC6021] data model to represent networks and topologies. The data model is divided into two parts. The first part of the model defines a network model that allows to define network hierarchies (i.e. network stacks) and to maintain an inventory of nodes contained in a network. The second part of the model augments the basic network model with information to describe topology information. Specifically, it adds the concepts of links and termination points to describe how nodes in a network are connected to each other. Moreover the model introduces vertical layering relationships between networks that can be augmented to cover both network inventories and network/service topologies.

The model can be augmented to describe specifics of particular types of networks and topologies. For example, an augmenting model can provide network node information with attributes that are specific to a particular network type. Examples of augmenting models include models for Layer 2 network topologies, Layer 3 network topologies, such as Unicast IGP, IS-IS [RFC1195] and OSPF [RFC2328], traffic engineering (TE) data [RFC3209], or any of the variety of transport and service topologies. Information specific to particular network types will be captured in separate, technology-specific models.

The basic data models introduced in this document are generic in nature and can be applied to many network and service topologies and inventories. The models allow applications to operate on an inventory or topology of any network at a generic level, where specifics of particular inventory/topology types are not required. At the same time, where data specific to a network type does come into play and the model is augmented, the instantiated data still adheres to the same structure and is represented in consistent fashion. This also facilitates the representation of network hierarchies and dependencies between different network components and network types.

The abstract (base) network YANG module introduced in this document, entitled "network.yang", contains a list of abstract network nodes and defines the concept of network hierarchy (network stack). The abstract network node can be augmented in inventory and topology models with inventory and topology specific attributes. Network hierarchy (stack) allows any given network to have one or more "supporting networks". The relationship of the base network model, the inventory models and the topology models is shown in the following figure (dotted lines in the figure denote possible augmentations to models defined in this document).

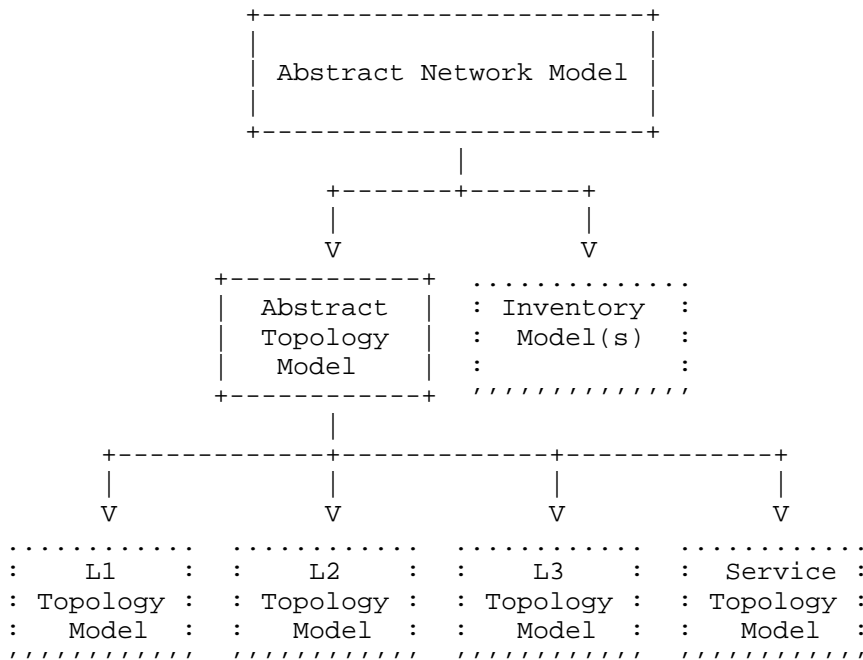


Figure 1: The network model structure

The network-topology YANG module introduced in this document, entitled "network-topology.yang", defines a generic topology model at its most general level of abstraction. The module defines a topology graph and components from which it is composed: nodes, edges and termination points. Nodes (from the network.yang module) represent graph vertices and links represent graph edges. Nodes also contain termination points that anchor the links. A network can contain multiple topologies, for example topologies at different layers and overlay topologies. The model therefore allows to capture relationships between topologies, as well as dependencies between nodes and termination points across topologies. An example of a topology stack is shown in the following figure.



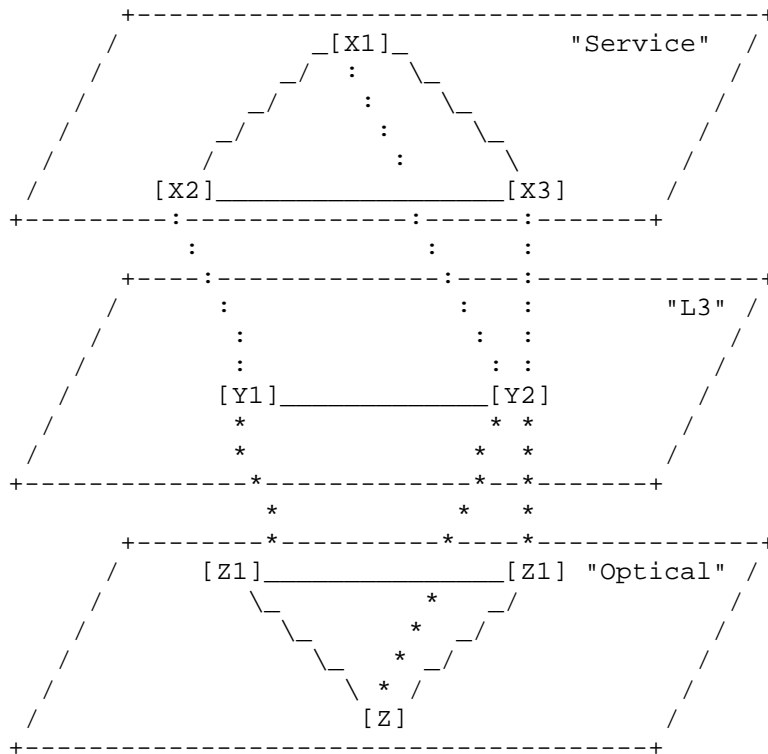


Figure 2: Topology hierarchy (stack) example

The figure shows three topology levels. At top, the "Service" topology shows relationships between service entities, such as service functions in a service chain. The "L3" topology shows network elements at Layer 3 (IP) and the "Optical" topology shows network elements at Layer 1. Service functions in the "Service" topology are mapped onto network elements in the "L3" topology, which in turn are mapped onto network elements in the "Optical" topology. The figure shows two Service Functions - X1 and X2 - mapping onto a single L3 network element; this could happen, for example, if two service functions reside in the same VM (or server) and share the same set of network interfaces. The figure shows a single "L3" network element mapped onto multiple "Optical" network elements. This could happen, for example, if a single IP router attaches to multiple ROADMs in the optical domain.

Another example of a service topology stack is shown in the following figure.

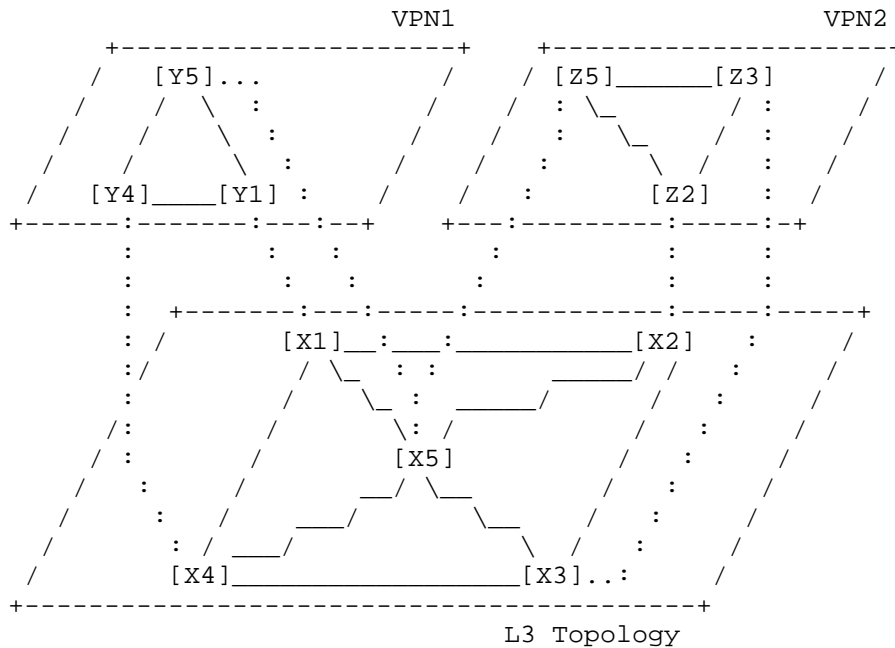


Figure 3: Topology hierarchy (stack) example

The figure shows two VPN service topologies (VPN1 and VPN2) instantiated over a common L3 topology. Each VPN service topology is mapped onto a subset of nodes from the common L3 topology.

There are multiple applications for such a data model. For example, within the context of I2RS, nodes within the network can use the data model to capture their understanding of the overall network topology and expose it to a network controller. A network controller can then use the instantiated topology data to compare and reconcile its own view of the network topology with that of the network elements that it controls. Alternatively, nodes within the network could propagate this understanding to compare and reconcile this understanding either among themselves or with help of a controller. Beyond the network element and the immediate context of I2RS itself, a network controller might even use the data model to represent its view of the topology that it controls and expose it to applications north of itself. Further use cases that the data model can be applied to are described in [topology-use-cases].

## 2. Definitions and Acronyms

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

**Data subtree:** An instantiated data node and the data nodes that are hierarchically contained within it.

**HTTP:** Hyper-Text Transfer Protocol

**IGP:** Interior Gateway Protocol

**IS-IS:** Intermediate System to Intermediate System protocol

**NETCONF:** Network Configuration Protocol

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

**URI:** Uniform Resource Identifier

**ReST:** Representational State Transfer, a style of stateless interface and protocol that is generally carried over HTTP

**YANG:** A data definition language for NETCONF

## 3. Model Structure Details

### 3.1. Base Network Model

The abstract (base) network model is defined in the `network.yang` module. Its structure is shown in the following figure. Brackets enclose list keys, "rw" means configuration data, "ro" means operational state data, and "?" designates optional nodes.

```

module: network
  +--rw network* [network-id]
    +--rw network-id          network-id
    +--ro server-provided?    boolean
    +--rw network-types
    +--rw supporting-network* [network-ref]
      | +--rw network-ref      leafref
    +--rw node* [node-id]
      +--rw node-id          node-id
      +--rw supporting-node* [network-ref node-ref]
        +--rw network-ref      leafref
        +--rw node-ref         leafref

```

Figure 4: The structure of the abstract (base) network model

The model contains a list of networks, contained underneath a root container for this module, "network". Each network is captured in its own list entry, distinguished via a network-id.

A network has a certain type, such as L2, L3, OSPF or IS-IS. A network can even have multiple types simultaneously. The type, or types, are captured underneath the container "network-types". In this module it serves merely as an augmentation target; network-specific modules will later introduce new data nodes to represent new network types below this target, i.e. insert them below "network-types" by ways of yang augmentation.

When a network is of a certain type, it will contain a corresponding data node. Network types SHOULD always be represented using presence containers, not leafs of empty type. This allows to represent hierarchies of network subtypes within the instance information. For example, an instance of an OSPF network (which, at the same time, is a layer 3 unicast IGP network) would contain underneath "network-types" another container "l3-unicast-igp-network", which in turn would contain a container "ospf-network".

A network can in turn be part of a hierarchy of networks, building on top of other networks. Any such networks are captured in the list "supporting-network". A supporting network is in effect an underlay network.

Furthermore, a network contains an inventory of nodes that are part of the network. The nodes of a network are captured in their own list. Each node is identified relative to its containing network by a node-id.

It should be noted that a node does not exist independently of a network; instead it is a part of the network that it is contained in.

In cases where the same entity takes part in multiple networks, or at multiple layers of a networking stack, the same entity will be represented by multiple nodes, one for each network. In other words, the node represents an abstraction of the device for the particular network that it is a part of. To represent that the same entity or same device is part of multiple topologies or networks, it is possible to create one "physical" network with a list of nodes for each of the devices or entities. This (physical) network, respectively the (entities) nodes in that network, can then be referred to as underlay network and nodes from the other (logical) networks and nodes, respectively. Note that the model allows to define more than one underlay network (and node), allowing for simultaneous representation of layered network- and service topologies and physical instantiation.

Similar to a network, a node can be supported by other nodes, and map onto one or more other nodes in an underlay network. This is captured in the list "supporting-node". The resulting hierarchy of nodes allows also to represent device stacks, where a node at one level is supported by a set of nodes at an underlying level. For example, a "router" node might be supported by a node representing a route processor and separate nodes for various line cards and service modules, a virtual router might be supported or hosted on a physical device represented by a separate node, and so on.

### 3.2. Base Network Topology Model

The abstract (base) network topology model is defined in the "network-topology.yang" module. It builds on the network model defined in the "network.yang" module, augmenting it with links (defining how nodes are connected) and termination-points (which anchor the links and are contained in nodes). The structure of the network topology module is shown in the following figure. Brackets enclose list keys, "rw" means configuration data, "ro" means operational state data, and "?" designates optional nodes.

```

module: network
  +--rw network* [network-id]
    +--rw network-id          network-id
    +--ro server-provided?    boolean
    +--rw network-types
    +--rw supporting-network* [network-ref]
      | +--rw network-ref      leafref
    +--rw node* [node-id]
      | +--rw node-id          node-id
      | +--rw supporting-node* [network-ref node-ref]
      | | +--rw network-ref      leafref
      | | +--rw node-ref         leafref
      | +--rw lnk:termination-point* [tp-id]
      | | +--rw lnk:tp-id          tp-id
      | | +--rw lnk:supporting-termination-point*
      | | | [network-ref node-ref tp-ref]
      | | +--rw lnk:network-ref      leafref
      | | +--rw lnk:node-ref         leafref
      | | +--rw lnk:tp-ref           leafref
    +--rw lnk:link* [link-id]
      +--rw lnk:link-id          link-id
      +--rw lnk:source
      | +--rw lnk:source-node      leafref
      | +--rw lnk:source-tp?      leafref
      +--rw lnk:destination
      | +--rw lnk:dest-node        leafref
      | +--rw lnk:dest-tp?        leafref
      +--rw lnk:supporting-link* [network-ref link-ref]
      | +--rw lnk:network-ref      leafref
      | +--rw lnk:link-ref         leafref

```

Figure 5: The structure of the abstract (base) network topology model

A node has a list of termination points that are used to terminate links. An example of a termination point might be a physical or logical port or, more generally, an interface.

Like a node, a termination point can in turn be supported by an underlying termination point, contained in the supporting node of the underlay network.

A link is identified by a link-id that uniquely identifies the link within a given topology. Links are point-to-point and unidirectional. Accordingly, a link contains a source and a destination. Both source and destination reference a corresponding node, as well as a termination point on that node. Similar to a node, a link can map onto one or more links in an underlay topology

(which are terminated by the corresponding underlay termination points). This is captured in the list "supporting-link".

### 3.3. Extending the model

In order to derive a model for a specific type of network, the base model can be extended. This can be done roughly as follows: for the new network type, a new YANG module is introduced. In this module a number of augmentations are defined against the network and network-topology YANG modules.

We start with augmentations against the network.yang module. First, a new network type needs to be defined. For this, a presence container that resembles the new network type is defined. It is inserted by means of augmentation below the network-types container. Subsequently, data nodes for any network-type specific node parameters are defined and augmented into the node list. The new data nodes can be defined as conditional ("when") on the presence of the corresponding network type in the containing network. In cases where there are any requirements or restrictions in terms of network hierarchies, such as when a network of a new network-type requires a specific type of underlay network, it is possible to define corresponding constraints as well and augment the supporting-network list accordingly. However, care should be taken to avoid excessive definitions of constraints.

Subsequently, augmentations are defined against network-topology.yang. Data nodes are defined both for link parameters, as well as termination point parameters, that are specific to the new network type. Those data nodes are inserted by way of augmentation into the link and termination-point lists, respectively. Again, data nodes can be defined as conditional on the presence of the corresponding network-type in the containing network, by adding a corresponding "when"-statement.

It is possible, but not required, to group data nodes for a given network-type under a dedicated container. Doing so introduces further structure, but lengthens data node path names.

In cases where a hierarchy of network types is defined, augmentations can in turn augmenting modules, with the module of a network "sub-type" augmenting the module of a network "super-type".

### 3.4. Discussion and selected design decisions

#### 3.4.1. Container structure

Rather than maintaining lists in separate containers, the model is kept relatively flat in terms of its containment structure. Lists of nodes, links, termination-points, and supporting-nodes, supporting-links, and supporting-termination-points are not kept in separate containers. Therefore, path specifiers used to refer to specific nodes, be it in management operations or in specifications of constraints, can remain relatively compact. Of course, this means there is no separate structure in instance information that separates elements of different lists from one another. Such structure is semantically not required, although it might enhance human readability in some cases.

#### 3.4.2. Underlay hierarchies and mappings

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

Where additional specifics about mappings between upper and lower layers are required, those can be captured in augmenting modules. For example, to express that a termination point in a particular network type maps to an interface, an augmenting module can introduce an augmentation to the termination point which introduces a leaf of type `ifref` that references the corresponding interface [RFC7223]. Similarly, if a node maps to a particular device or network element, an augmenting module can augment the node data with a leaf that references the network element.

It is possible for links at one level of a hierarchy to map to multiple links at another level of the hierarchy. For example, a VPN topology might model VPN tunnels as links. Where a VPN tunnel maps to a path that is composed of a chain of several links, the link will contain a list of those supporting links. Likewise, it is possible for a link at one level of a hierarchy to aggregate a bundle of links at another level of the hierarchy.

#### 3.4.3. Use of groupings

The model makes use of groupings, instead of simply defining data nodes "in-line". This allows to more easily include the corresponding data nodes in notifications, which then do not need to respecify each data node that is to be included. The tradeoff for this is that it makes the specification of constraints more complex,



because constraints involving data nodes outside the grouping need to be specified in conjunction with a "uses" statement where the grouping is applied. This also means that constraints and XPath-statements need to be specified in such a way that they navigate "down" first and select entire sets of nodes, as opposed to being able to simply specify them against individual data nodes.

#### 3.4.4. Cardinality and directionality of links

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

#### 3.4.5. Multihoming and link aggregation

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

#### 3.4.6. Mapping redundancy

In a hierarchy of networks, there are nodes mapping to nodes, links mapping to links, and termination points mapping to termination points. Some of this information is redundant. Specifically, if the link-to-links mapping is known, and the termination points of each link are known, termination point mapping information can be derived via transitive closure and does not have to be explicitly configured. Nonetheless, in order to not constrain applications regarding which mappings they want to configure and which should be derived, the model does provide for the option to configure this information explicitly. The model includes integrity constraints to allow for validating for consistency.

### 3.4.7. Typing

A network's network types are represented using a container which contains a data node for each of its network types. A network can encompass several types of network simultaneously, hence a container is used instead of a case construct, with each network type in turn represented by a dedicated presence container itself. The reason for not simply using an empty leaf, or even simpler, do away even with the network container and just use a leaf-list of network-type instead, is to be able to represent "class hierarchies" of network types, with one network type refining the other. Network-type specific containers are to be defined in the network-specific modules, augmenting the network-types container.

### 3.4.8. Representing the same device in multiple networks

One common requirement concerns the ability to represent that the same device can be part of multiple networks and topologies. However, the model defines a node as relative to the network that it is contained in. The same node cannot be part of multiple topologies. In many cases, a node will be the abstraction of a particular device in a network. To reflect that the same device is part of multiple topologies, the following approach might be chosen: A new type of network to represent a "physical" (or "device") network is introduced, with nodes representing devices. This network forms an underlay network for logical networks above it, with nodes of the logical network mapping onto nodes in the physical network.

This scenario is depicted in the following figure. It depicts three networks with two nodes each. A physical network P consists of an inventory of two nodes, D1 and D2, each representing a device. A second network, X, has a third network, Y, as its underlay. Both X and Y also have the physical network P as underlay. X1 has both Y1 and D1 as underlay nodes, while Y1 has D1 as underlay node. Likewise, X2 has both Y2 and D2 as underlay nodes, while Y2 has D2 as underlay node. The fact that X1 and Y1 are both instantiated on the same physical node D1 can be easily derived.

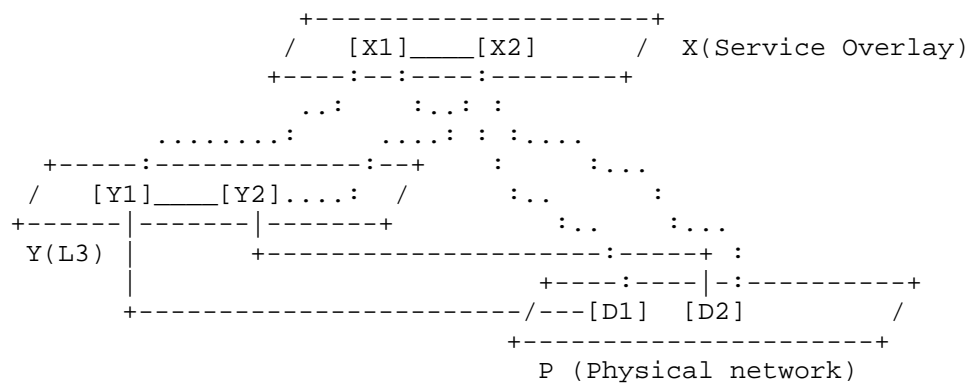


Figure 6: Topology hierarchy example - multiple underlays

In the case of a physical network, nodes represent physical devices and termination points physical ports. It should be noted that it is also conceivable to augment the model for a physical network-type, defining augmentations that have nodes reference system information and termination points reference physical interfaces, in order to provide a bridge between network and device models.

### 3.5. Items for further discussion

YANG requires data needs to be designated as either configuration or operational data, but not both, yet it is important to have all network information, including vertical cross-network dependencies, captured in one coherent model. In most cases network topology information is discovered about a network; the topology is considered a property of the network that is reflected in the model. That said, it is conceivable that certain types of topology need to also be configurable by an application.

There are several alternatives in which this can be addressed. The alternative chosen in this draft does not restrict network topology information as read-only, but includes a flag that indicates for each network whether it should be considered as read-only or configurable by applications.

An alternative would be to designate network list elements as read only. The read-only network list includes each network; it is the complete reference. In parallel a second network list is introduced. This list serves the purpose of being able to configure networks which are then mirrored in the read-only list. The configurable network list adheres to the same structure and uses the same groupings as its read-only counterpart. As most data is defined in those groupings, the amount of additional definitions required will

be limited. A configurable network will thus be represented twice: once in the read-only list of all networks, a second time in a configuration sandbox.

Similar considerations apply to scenarios in which data is subject to configuration, but implementations want to be smart enough require only some mapping information, such as which link is supported by which other links, while automatically deriving other mapping information where possible, such as which termination points are supported by which underlay termination points. To accommodate such cases, separate provision may again be made by including another "server-provided" option.

#### 4. YANG Modules

##### 4.1. Defining the Abstract Network: network.yang

```
<CODE BEGINS> file "network.yang"
module network {
  yang-version 1;
  namespace "urn:TBD:params:xml:ns:yang:nodes";
  prefix nd;

  import ietf-inet-types { prefix inet; }

  organization "TBD";
  contact
    "WILL-BE-DEFINED-LATER";
  description
    "This module defines a common base model for a collection
    of nodes in a network. Node definitions s are further used
    in network topologies and inventories.";

  revision 2014-3-9 {
    description
      "Initial revision.";
    reference "draft-clemm-i2rs-yang-network-topo-04";
  }

  typedef node-id {
    type inet:uri;
  }

  typedef network-id {
    type inet:uri;
  }

  grouping network-ref {
```

```
    leaf network-ref {
      type leafref {
        path "/network/topology-id";
      }
    }
  }

  grouping node-ref {
    uses network-ref;
    leaf node-ref {
      type leafref {
        path "/network[network-id=current()] +
              '/../network-ref]/node/node-id";
      }
    }
  }
}

list network {
  key "network-id";

  leaf network-id {
    type network-id;
  }

  leaf server-provided {
    type boolean;
    config false;
  }

  container network-types {
  }

  list supporting-network {
    key "network-ref";
    leaf network-ref {
      type leafref {
        path "/network/network-id";
      }
    }
  }
}

list node {
  key "node-id";
  leaf node-id {
    type node-id;
  }
  list supporting-node {
    key "network-ref node-ref";
  }
}
```

```

        leaf network-ref {
            type leafref {
                path "../../supporting-network/network-ref";
            }
        }
        leaf node-ref {
            type leafref {
                // path "/network[network-id=current()] +
                // "/../network-ref]/node/node-id";
                path "/network/node/node-id";
            }
        }
    }
}
}
}
}
}
<CODE ENDS>

```

#### 4.2. Creating Abstract Network Topology: network-topology.yang

```

<CODE BEGINS> file "network-topology.yang"
module network-topology {
    yang-version 1;
    namespace "urn:TBD:params:xml:ns:yang:links";
    prefix lnk;

    import ietf-inet-types { prefix inet; }
    import nodes { prefix nd; }

    organization "TBD";
    contact
        "WILL-BE-DEFINED-LATER";
    description
        "This module defines a common base model for a collection of links
        connecting nodes.";

    revision 2014-3-9 {
        description
            "Initial revision.";
        reference "draft-clemm-i2rs-yang-network-topo-04";
    }

    typedef link-id {
        type inet:uri;
        description
            "An identifier for a link in a topology.
            The identifier may be opaque.

```

```

    The identifier SHOULD be chosen such that the same link in a
    real network topology will always be identified through the
    same identifier, even if the model is instantiated in separate
    datastores. An implementation MAY choose to capture semantics
    in the identifier, for example to indicate the type of link
    and/or the type of topology that the link is a part of.";
}

typedef tp-id {
    type inet:uri;
    description
        "An identifier for termination points on a node.
        The identifier may be opaque.
        The identifier SHOULD be chosen such that the same TP in a
        real network topology will always be identified through the
        same identifier, even if the model is instantiated in separate
        datastores. An implementation MAY choose to capture semantics
        in the identifier, for example to indicate the type of TP
        and/or the type of node and topology that the TP is a part
        of.";
}

grouping link-ref {
    description
        "A type for an absolute reference a link instance.
        (This type should not be used for relative references.
        In such a case, a relative path should be used instead.)";
    uses nd:network-ref;
    leaf link-ref {
        type leafref {
            path "/nd:network[nd:network-id=current()" +
                "../nd:network-ref]/link/link-id";
        }
    }
}

grouping tp-ref {
    description
        "A type for an absolute reference to a termination point.
        (This type should not be used for relative references.
        In such a case, a relative path should be used instead.)";
    uses nd:node-ref;
    leaf tp-ref {
        type leafref {
            path "/nd:network[nd:network-id=current()" +
                "../nd:network-ref]/nd:node[nd:node-id=current()" +
                "../nd:node-ref]/termination-point/tp-id";
        }
    }
}

```

```

    }
  }

  augment "/nd:network" {
    list link {
      key "link-id";
      leaf link-id {
        type link-id;
        description
          "The identifier of a link in the topology.
           A link is specific to a topology to which it belongs.";
      }
      description
        "A Network Link connects a by Local (Source) node and
         a Remote (Destination) Network Nodes via a set of the
         nodes' termination points.
         As it is possible to have several links between the same
         source and destination nodes, and as a link could
         potentially be re-homed between termination points, to
         ensure that we would always know to distinguish between
         links, every link is identified by a dedicated link
         identifier.
         Note that a link models a point-to-point link, not a
         multipoint link.
         Layering dependencies on links in underlay topologies are
         not represented as the layering information of nodes and of
         termination points is sufficient.";
      container source {
        description
          "This container holds the logical source of a particular
           link.";
        leaf source-node {
          type leafref {
            path "../.../nd:node/nd:node-id";
          }
          mandatory true;
          description
            "Source node identifier, must be in same topology.";
        }
        leaf source-tp {
          type leafref {
            path "../.../nd:node[nd:node-id=current()/.." +
              "/source-node]/termination-point/tp-id";
          }
          description
            "Termination point within source node that terminates
             the link.";
        }
      }
    }
  }

```



```

    }
    container destination {
        description
            "This container holds the logical destination of a
            particular link.";
        leaf dest-node {
            type leafref {
                path "../..../nd:node/nd:node-id";
            }
            mandatory true;
            description
                "Destination node identifier, must be in the same
                network.";
        }
        leaf dest-tp {
            type leafref {
                path "../..../nd:node[nd:node-id=current()/.." +
                    "/dest-node]/termination-point/tp-id";
            }
            description
                "Termination point within destination node that
                terminates the link.";
        }
    }
}
list supporting-link {
    key "network-ref link-ref";
    description
        "Identifies the link, or links, that this link
        is dependent on.";
    leaf network-ref {
        type leafref {
            path "../..../nd:supporting-network/nd:network-ref";
        }
        description
            "This leaf identifies in which underlay topology
            supporting link is present.";
    }
    leaf link-ref {
        type leafref {
            path "/nd:network[nd:network-id=" +
                "current()/../network-ref]/link/link-id";
        }
        description
            "This leaf identifies a link which is forms a part
            of this link's underlay. Reference loops, where
            a link identifies itself as its underlay, either
            directly or transitively, are not allowed.";
    }
}

```

```

    }
  }
}

augment "/nd:network/nd:node" {
  list termination-point {
    key "tp-id";
    description
      "A termination point can terminate a link.
      Depending on the type of topology, a termination point
      could, for example, refer to a port or an interface.";
    leaf tp-id {
      type tp-id;
      description
        "Termination point identifier.";
    }
  }
  list supporting-termination-point {
    key "network-ref node-ref tp-ref";
    description
      "The leaf list identifies any termination points that
      the termination point is dependent on, or maps onto.
      Those termination points will themselves be contained
      in a supporting node.
      This dependency information can be inferred from
      the dependencies between links. For this reason,
      this item is not separately configurable. Hence no
      corresponding constraint needs to be articulated.
      The corresponding information is simply provided by the
      implementing system.";
    leaf network-ref {
      type leafref {
        path "../..../nd:supporting-node/nd:network-ref";
      }
      description
        "This leaf identifies in which topology the
        supporting termination point is present.";
    }
    leaf node-ref {
      type leafref {
        path "../..../nd:supporting-node/nd:node-ref";
      }
      description
        "This leaf identifies in which node the supporting
        termination point is present.";
    }
    leaf tp-ref {
      type leafref {
        path "/nd:network[nd:network-id=" +

```

```
        "current()/../network-ref]/nd:node[nd:node-id=" +  
        "current()/../node-ref]/termination-point" +  
        "/tp-id";  
    }  
    description  
        "Reference to the underlay node, must be in a  
        different topology";  
    }  
    }  
    }  
    }  
}
```

<CODE ENDS>

## 5. Security Considerations

The transport protocol used for sending the topology data **MUST** support authentication and **SHOULD** support encryption. The data-model by itself does not create any security implications.

## 6. Contributors

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

- o Ken Gray, Cisco Systems
- o Tom Nadeau, Brocade
- o Aleksandr Zhdankin, Cisco

## 7. Acknowledgements

We wish to acknowledge the helpful contributions, comments, and suggestions that were received from Alia Atlas, Vishna Pavan Beeram, Andy Bierman, Martin Bjorklund, Igor Bryskin, Benoit Claise, Susan Hares, Xufeng Liu, Ladislav Lhotka, Carlos Pignataro, Juergen Schoenwaelder, and Xian Zhang.

## 8. References

### 8.1. Normative References

- [RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, December 1990.

- [RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, April 1998.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, May 2014.

## 8.2. Informative References

- [restconf] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", I-D draft-ietf-netconf-restconf-04, January 2015.
- [topology-use-cases] Medved, J., Previdi, S., Lopez, V., and S. Amante, "Topology API Use Cases", I-D draft-amante-i2rs-topology-use-cases-01, October 2013.
- [yang-json] Lhotka, L., "JSON Encoding of Data Modeled with YANG", I-D draft-ietf-netmod-yang-json-03, February 2015.

## Authors' Addresses

Alexander Clemm  
Cisco

EMail: alex@cisco.com

Jan Medved  
Cisco

EMail: jmedved@cisco.com

Robert Varga  
Cisco

EMail: rovarga@cisco.com

Tony Tkacik  
Cisco

EMail: ttkacik@cisco.com

Nitin Bahadur  
Bracket Computing

EMail: nitin\_bahadur@yahoo.com

Hariharan Ananthakrishnan  
Packet Design

EMail: hari@packetdesign.com

I2RS Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 26, 2015

S. Hares  
L. Wang  
S. Zhuang  
Huawei  
October 23, 2014

An I2RS BGP Information Model  
draft-hares-i2rs-bgp-im-02.txt

Abstract

This document introduces an information model for i2RS BGP protocol and network policy that aligns with the I2RS BGP use cases. This draft utilizes the general Policy based routing structured found in the RIB Information Model (IM) and Policy Base Policy.

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 April 26, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	2
2. Definitions and Acronyms . . . . .	3
3. BGP Data . . . . .	3
3.1. BGP Protocol . . . . .	3
3.1.1. Yang Data Model for I2RS BGP IM Protocol Block . . . . .	4
3.1.2. Elements in I2RS Protocol Block . . . . .	5
3.2. bgp_ipv4_instance . . . . .	8
3.2.1. Yang Data Model bgp ipv4 unicast instance . . . . .	8
3.2.2. Elements in bgp ipv4 unicast instance . . . . .	8
3.3. bgp local rib . . . . .	11
3.3.1. bgp-local-rib for IPv4 routes . . . . .	12
3.4. BGP Peer . . . . .	16
3.4.1. Yang Top Level for BGP Peer . . . . .	16
3.4.2. Elements in BGP peer Structure . . . . .	17
3.4.3. Reading/Writing Policy in I2RS IM model for BGP Peer . . . . .	20
3.5. Additional variable used within all RIBS . . . . .	21
4. BGP Yang top-level description . . . . .	22
5. IANA Considerations . . . . .	87
6. Security Considerations . . . . .	87
7. Informative References . . . . .	88
Authors' Addresses . . . . .	89

## 1. Introduction

The Interface to the Routing System (I2RS) provides read and write access to the information and state within the routing process within routing elements via protocol message exchange between an I2RS Client and an I2RS Agent associated with the a routing system. The [I-D.ietf-i2rs-architecture] describes the basic interactions of this exchange. One of the important functions of this messages exchange is for the I2RS client to interact with one or more I2RS agents to collect information from network routing systems.

One of the key information pieces that I2RS clients may collect from I2RS agents is the BGP information regarding BGP peers, BGP routes, and network topologies created within BGP. This document introduces a basic information model for BGP policies. This policy model can be linked with other information models such as the I2RS RIB informational model [I-D.ietf-i2rs-rib-info-model], and uses the generic policy found in the Policy Routing modules in [I-D.hares-i2rs-info-model-policy]. This basic BGP Information Model is an extendable policy model is a product of the industry approach to I2RS.

## 2. Definitions and Acronyms

BGP: Border Gateway Protocol

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

NETCONF: The Network Configuration Protocol as defined in [RFC6536]

RESTCONF: The REST-like protocol that provides a programmatic interface over HTTP for accessing data defined in YANG, using datastores defined in NETCONF Protocol [I-D.ietf-netconf-restconf] as defined in RBNF: Routing Backus-Naur Form [RFC5511].

## 3. BGP Data

BGP data includes information related to the BGP protocol instances, BGP Peer information, BGP Route information, and BGP policy. This section describes the information associated with the I2RS agent accessible via BGP.

### 3.1. BGP Protocol

BGP protocol support within a router (or virtual routing device) may have multiple instances of the BGP protocol. This section describes the information regarding the BGP protocol available to the I2RS Agent. The section begins with a top-level diagram of the Yang Data Model, and then goes to define the Yang data model in text.

#### Protocol

The BGP protocol information stores a group of information associated with a BGP protocol implementation. This group of informational includes:

- \* Router-ID (R/W) - Router-ID for BGP protocol,
- \* AS(R/W)- Autonomous System Number,
- \* protocol\_status (R) - administrative status of the protocol,
- \* shutdown\_protocol (R/W) - Shutdown the BGP protocol,
- \* ROLES (R) - a composite set of roles BGP peers play,
- \* bgp-ipv4-uni-instance (R/W) - instance for IPv4 unicast



- \* `bgp-labeled-ipv4instance` (R/W) - instance for IPv4 unicast with labels
- \* `bgp-ipv4-multi-instance` (R/W) - instance for IPv4 multicast
- \* `bgp-ipv4-vpn-instance` (R/W) - instance for IPv4 VPNs
- \* `bgp-ipv4-l3vpn-instance-list` (R/W) list of L3VPN instances indexed by the BGP instance name
- \* `bgp-ipv6-uni-instance` (R/W) - instance handling IPv6 unicast
- \* `bgp-ipv6-label-instance` (R/W) - instance with IPv6 prefix plus labels
- \* `bgp-ipv6-multi-instance` (R/W) - BGP IPv6 multicast instance
- \* `bgp-ipv6-vpn-instance-list` (R/W) - list of IPv6 L3VPN instances

The sections below describe these elements.

#### 3.1.1. Yang Data Model for I2RS BGP IM Protocol Block

```

+--rw bgp-protocol
  +--rw router-id?                               inet:ip-address
  +--rw as-number?                               uint32
  +--ro protocol-status?                         enumeration
  +--rw shutdown-protocol?                       boolean
  +--ro bgp-role?                                enumeration
  +--rw bgp-ipv4-uni-instance
  | ...
  +--rw bgp-labeled-ipv4-instance
  | ...
  +--rw bgp-ipv4-multi-instance
  | ...
  +--rw bgp-vpnv4-instance
  | ...
  +--rw bgp-ipv4-l3vpn-instance-list* [bgp-l3vpn-instance-name]
  | ...
  +--rw bgp-ipv6-uni-instance
  | ...
  +--rw bgp-labeled-ipv6-instance
  | ...
  +--rw bgp-ipv6-multi-instance
  | ...
  +--rw bgp-vpnv6-instance
  | ...
  +--rw bgp-ipv6-l3vpn-instance-list* [bgp-l3vpn-instance-name]
  | ...

```

### 3.1.2. Elements in I2RS Protocol Block

This section describes each element of the I2RS Protocol Block.

#### Router-ID (R)

This I2RS BGP IM element stores the ROUTER-ID which is configured in the BGP protocol as the default router-id. Instances of a protocol may configure unique router-IDs. The router-ID is a 32 bit unsigned value imported from the running BGP process.

#### Router-ID (W)

Upon a write from the I2RS Client to this I2RS BGP IM element, this I2RS Agent will temporarily change this default in the BGP protocol. I2RS Agent-routing system software must be able to handle both the original value (from node configuration systems), and the I2RS value as valid defaults. This change in the ROUTER-ID default is passed to the BGP instances spawned from the BGP

protocol. After the I2RS Client stops the relationship with the I2RS Agent, the Router-ID value reverts to the configured value.

#### AS(R/W)

This I2RS BGP IM element stores the default AS number configured in the BGP implementation. This value is 8 byte value represented as a decimal value. This value is a read-write value. The initial value of the AS is imported by the I2RS Agent from the BGP protocol in the BGP routing system. This the normal case. If this data element is written to, I2RS stores the default AS number value in the BGP protocol as the new default AS in the BGP protocol. This does not change the default AS number in existing BGP instances with the existing peers. New BGP instances spawned after the I2RS changes the default AS value, should have the I2RS specified default AS. Upon the completion of the I2RS client-I2RS Agent association, the default AS value should be returned to the configured value.

#### protocol\_status (R)

This I2RS BGP IM element stores an administrative status for the protocol activity of "no-shutdown" (0x00) or "shutdown" (0x01). If the routing entity the I2RS protocol is attached has BGP "shutdown", no data will be available from any BGP instances. This is a read-only variable.

#### shutdown\_protocol(R/W)

A write of "TRUE" to the "shutdown\_protocol" status in the I2RS Agent indicates the I2RS agent should request a shutdown of the BGP protocol from the routing function. The default value is "FALSE" for this I2RS variable. The read of the "shutdown\_protocol" indicate the current status of the "shutdown\_protocol" variable in the I2RS Agent. This is different than the "protocol\_status" which indicates the status of the protocol: shutdown or not "shutdown" (running).

#### bgp\_role (R)

The BGP\_ROLE provides a list of BGP Roles found within the BGP Instances associated with the BGP protocol function on the device associated with the I2RS agent. These can be ASBR (AS border router E-BGP), PE (Provider Edge E-BGP), IBGP peer, or RR (Route Reflector).

#### bgp-ipv4-uni-instance (R-W)

This element BGP instances provides support for IPv4 unicast , and contains the associated bgp local rib, bgp peer list, bgp rib-in, and bgp-rib-out associated with IPv4 Unicast.

bgp-labeled-ipv4-instance (R-W)

This element provides the bgp instance with routes associated with label routes. (Contains bgp local rib, bgp peer list, bgp rib-in, and bgp rib-out for these routes).

bgp-ipv4-multi-instance (R-W)

This element contains an bgp instance for IPv4 multicast. (Contains bgp local rib, bgp peer list, bgp rib-in, and bgp rib-out for these routes).

bgp-vpnv4-instance (R-W)

This element contains BGP instance IPv4 VPNs. (Contains bgp local rib, bgp peer list, bgp rib-in, and bgp rib-out for these routes).

bgp-ipv4-L3vpn-instance-list\* (R-W)

This element contains BGP instance IPv4 VPNs. (Contains bgp local rib, bgp peer list, bgp rib-in, and bgp rib-out for these routes).

bgp-ipv6-uni-instance (R-W)

This element contains BGP instance IPv6 unicast routes. (Contains bgp local rib, bgp peer list, bgp rib-in, and bgp rib-out for these routes).

bgp-labeled-ipv6-instance (R-W)

This element provides the bgp instance with routes associated with label routes. (Contains bgp local rib, bgp peer list, bgp rib-in, and bgp rib-out for these routes).

bgp-ipv6-multi-instance(R-W)

This element contains an bgp instance for IPv6 multicast. (Contains bgp local rib, bgp peer list, bgp rib-in, and bgp rib-out for these routes).

bgp-vpnv6-instance (R-W)

This element contains BGP instance IPv6 VPNs. (Contains bgp local rib, bgp peer list, bgp rib-in, and bgp rib-out for these routes).

bgp-ipv6-L3vpn-instance-list\* (R-W)

This element contains BGP instance IPv4 VPNs. (Contains bgp local rib, bgp peer list, bgp rib-in, and bgp rib-out for these routes).

### 3.2. bgp\_ipv4\_instance

BGP protocol support within a router(or virtual routing device) may have multiple instances of the BGP protocol. This section describes the information regarding the BGP protocol available to the I2RS Agent. The section begins with a top-level diagram of the Yang Data Model, and then goes to define the Yang data model in text.

#### 3.2.1. Yang Data Model bgp ipv4 unicast instance

This figure below the yang model at bgp instance to handle the bgp instance. Within Yang, the definition is based on a a grouping which has name, I2rs creation flags, standard instance type, vendor type, afi, safi, bgp local RIB, and BGP peers. The standard instances types are: IPv4 unicast, IPv4 multicast, IPv4 L3vpn, VPLS, MDT, EVPN, BGP-ls, IPv4-mpls-vpn, IPv4-mpls-mvpn, BGP route-targets, IPv4 flow, IPv4 vpn-flow, IPv4 vrf-list, IPv6 vrf-list, and IPv6 unit, and IPv6 vpnv6. It also could contain instance types defined by vendors (bgp-vendor-type). This informational model contains the details of the instance grouping and its subcomponents: bgp-instance-name, bgp-instance-create, bgp-instance-type bgp-vendor-type, afi, safi, bgp-local-rib, and bgp-peer list. It does not enumerate these for each of the instance types, but provides a full yang tree of this definition in section x.

```

+--rw bgp-ipv4-uni-instance
|   +--rw bgp-instance-name?      string
|   +--rw bgp-instance-create?    enumeration
|   +--rw bgp-instance-type?      enumeration
|   +--rw bgp-vendor-type?        enumeration
|   +--rw afi?                    afi-def
|   +--rw safi?                   safi-def
|   +--rw bgp-local-rib
|
+--rw bgp-peer-list* [bgp-peer-name]

```

#### 3.2.2. Elements in bgp ipv4 unicast instance

The definitions of the fields in BGP Instance are:

BGP\_INSTANCE\_NAME (R/W)

The BGP\_INSTANCE\_NAME provides a name to uniquely identify the BGP instance.

#### BGP\_INSTANCE\_CREATION(R/W)

is the flag to indicate whether the I2RS client is requesting the I2RS agent to have the BGP routing protocol create a new BGP instance for these I2RS parameters. The internal status includes:

- \* Not I2RS created (NOT-I2RS),
- \* I2RS client requests creation (I2RSCLIENT\_CREATE\_BGP\_INSTANCE),
- \* I2RS Agent rejects creation (I2RSAGENT\_REJECTS\_BGP\_INSTANCE\_CREATE),
- \* I2RS Agent attempting creation (I2RSAGENT\_ATTEMPTS\_BGP\_INSTANCE\_CREATE), and
- \* I2RS Agent failed creation (I2RSAGENT\_FAILS\_BGP\_INSTANCE\_CREATE)
- \* I2RS Agent created BGP instance (I2RSAGENT\_CREATED\_BGP\_INSTANCE)

#### BGP\_INSTANCE\_TYPE (R/W)

contains flag field that summarizes common AFI-SAFIs within a flag word. This enumeration

- \* BGP\_IPV4\_UNI flag - is a flag for AFI of IPV4 and a SAFI of Unicast (1).
- \* BGP\_IPV4\_MULTI flag - is a flag for AFI of IPV4 and a SAFI Multicast (2).
- \* BGP\_L3VPN I is a flag for AFI of IPv4 and a SAFI of MPLS labels (4).
- \* BGP\_IPv4\_MVPN- is a flag for AFI of IPv4 and a SAFI of MCAST VPN (5).
- \* BGP\_VPLS - is a flag for an AFI of IPv4 and a SAFI VPLS (65).
- \* BGP\_IPv4\_MDT - is a flag for AFI of IPv4 and a SAFI of BGP MDT (66).
- \* BGP\_EVPN - is a flag for APFI of IPv4 and SAFI of EVPNs (70)

- \* BGP\_LS - is a flag for an AFI of IPv4 and a SAFI of BGP link state (LS) (71).
- \* BGP\_IPV4\_MPLS\_VPN - is a flag for an AFI of IPv4 and a SAFI MPLS-labeled Unicast VPNs (128).
- \* BGP\_IPv4\_MPLS\_MVPN - is a flag for an AFI of IPv4 and a SAFI MPLS-labeled Multicast VPNs (129).
- \* BGP\_IPv4\_ROUTE\_TARGET - is a flag for the AFI of IPv4 and a SAFI of Route Target Constraints (132)
- \* BGP\_IPv4\_FLOW - is a flag for an AFI of IPv4 and a SAFI of IPv4 dissemination of flow specification rules (133).
- \* BGP\_IPv4\_VPN\_FLOW - is a flag for an AFI of IPv4 and a SAFI of IPv4 VPN dissemination of flow specification rules(134).
- \* BGP\_L2AD - is a flag for an AFI of IPv4 and a SAFI L2 Auto-discovery (140)
- \* BGP\_IPv6\_UNI - is a flag for AFI of IPv6 and SAFI of Unicast (1)
- \* BGP\_IPv6\_MULTI - is a flag for API of IPv65 and a SAFI of Multicast (2)
- \* BGP\_IPV6\_MPLS\_VPN - is a flag for AFI of IPV4 and SAFI of Unicast MPLS VPNS.

#### BGP\_VENDOR\_TYPE(R/W)

contains a field to specify Vendor to identify different mechanism of BGP configuration.

#### AFI (R/W)

The AFI value from the standard values found in IANA's Address Family Numbers.

#### SAFI (R/W)

The AFI value from the standard values found in IANA's SAFI values.

#### bgp-local-rib (R/W)

The `bgp-local-rib` contains the contains a local RIB for BGP associated with this BGP instance. Every BGP instance has its own local RIB which contains the routes received by all Peers (`Adj-RIB-IN` per Peer), and selected by policy as the active route by the local BGP speaker's Decision Process.

`bgp-peer-list` (R/W)

contains a list of BGP Peer information (`BGP_PEER`) associated with this BGP instance. A write to this BGP Peer list adds a BGP Peer to this BGP instance.

`rib-in` (R/W)

This contains `bgp-rib-in` route list.

`rib-out` (R/W)

This contains `bgp-rib-out` route list.

### 3.3. `bgp local rib`

The routes associated with each BGP instance are the BGP local RIB, the `AdjRIBIn`, and the `AdjRIBOut`. The BGP local RIB is described below. The `AdjRIBIn` is the `bgp_rib_in` described in the `BGP_Peer` section. The policy that sets preferences on the routes received by the peer is identified by the `Peer_policy_in` (an policy set name), and fully readable by the `Peer_policy_in_pset` (policy set pointer) entry.

The `AdjRIBOut` is contained in `bgp_rib_out` (a route list) described in the BGP Peer Section. The Policy associated with the `AdjRIBOut` is identified by the `Peer_policy_out` (a policy set name) and fully readable by the `Peer_policy_out_pset` (policy set pointer).

This section describes the `bgp local ribs` for IPv4 unicast routes as a prototype for all rib routes.

Each `bgp local RIB` listed below may be created and stored as a prefix tree. However, yang model descriptions describe it as a list of routes with an index of prefix and length. The above top-level yang demonstrates this. This yang mechanism requires a unique definition per AFI/SAFI.

Each rib is indexed by the prefix/prefix-length and contains route type, attribute list, route creation information, and state information. The IPv4 local RIB shows the full definition of the



route type, attribute list, route creation, and state information. Subsequent RIBs do not repeat this information.

### 3.3.1. bgp-local-rib for IPv4 routes

This section contains the overview of the bgp-local-rib for IPv4 routes, and serves as a prototype for all other ribs.

#### 3.3.1.1. Yang Top-level for bgp-local-rib for IPv4 routes

This section describes the high-level yang definition for a local RIB with IPv4 routes. This serves as an prototype of all bgp-local-ribs.

```

+--rw bgp-local-rib
|   +--rw bgp-route-list* [ipv4-route ipv4-prefix-length]
|   |   +--rw ipv4-route                inet:ipv4-prefix
|   |   +--rw ipv4-prefix-length        uint8
|   |   +--rw bgp-route-type?           enumeration
|   |   +--rw route-admin-distaince    uint16
|   |   +--rw bgp-attribute-list
|   |   |   +--rw bgp-origin?           enumeration
|   |   |   +--rw bgp-aspath
|   |   |   |   +--rw usascount?        uint16
|   |   |   |   +--rw ulrefcount?      uint16
|   |   |   |   +--rw asstring?        string
|   |   |   |   +--ro usascountfrcomp? uint16
|   |   |   |   +--rw usastotalcount?  uint16
|   |   |   |   +--rw usas4pathlen?    uint16
|   |   |   |   +--rw as4pathvalue?    string
|   |   |   +--rw bgp-nexthop?         inet:ip-address
|   |   |   +--rw bgp-med?              uint32
|   |   |   +--rw bgp-localpref?       uint32
|   |   |   +--rw bgp-atomic-aggregate? uint32
|   |   |   +--rw bgp-aggregator
|   |   |   |   +--rw ulipaddress?      uint32
|   |   |   |   +--rw ulasnumber?      uint32
|   |   |   +--rw bgp-commattr
|   |   |   |   +--rw uscommsize?       uint16
|   |   |   |   +--rw ulrefcomm?       uint32
|   |   |   |   +--rw commattr-value?  string
|   |   |   +--rw bgp-extcommattr
|   |   |   |   +--rw custome-community
|   |   |   |   |   +--rw valid         boolean
|   |   |   |   |   +--rw insertion-pont uint32
|   |   |   |   |   +--rw community-id  uint8
|   |   |   |   |   +rw cost-id         uint32
|   |   |   |   +--rw usextcommsize?    uint16
|   |   |   |   +--rw ulrefcount?       uint32
|   |   |   |   +--rw extcommattr-value? string
|   |   |   +--rw bgp-clusterlist
|   |   |   |   +--rw uscluslen?        uint16
|   |   |   |   +--rw ulrefclus?       uint32
|   |   |   |   +--rw clusterlist-value? string
|   |   |   +--rw bgp-originator-id?   uint32
|   |   +--ro bgp-route-create?         enumeration
|   |   +--ro bgp-rt-state-info
|   |   |   +--ro rib-current-state?     rib-state-def
|   |   |   +--ro rib-last-state?       rib-state-def
|   |   |   +--ro rib-rejected-reason?   enumeration
|   |   |   +--ro not-preferred-reason?  enumeration

```

### 3.3.1.2. Common definitions for bgp-local-rib for IPv4 routes

The BGP Local RIB Entries are:

BGP\_ROUTE\_TYPE(R/W)

contains a ROUTE TYPE indicator (enumerated type). The enumerated types are:

- \* bgp-route-type-rd - route with RD information
- \* bgp-route-type-ipv4- IPv4 unicast route
- \* bgp-route-type-ipv6 -IPv6 unicast route
- \* bgp-route-labeled-ipv4 - IPv4 with labels attached
- \* bgp-route-labeled-ipv6 - IPve6 associated labels
- \* bgp-route-type-flow - flow specifications
- \* bgp-route-type-evpn - EVPN API/SAFI
- \* bgp-route-type-mvpn - Mutlicast VPN
- \* bgp-route-type-vpls - VPLS routes
- \* bgp-route-type-bgp-ls - BGP Link-state distribution info
- \* bgp-route-type-l2vpn-signaling-nlri - L2VPN signalling NLRI
- \* bgp-rt-constraint - constraint RT AFI/SAFI
- \* pw-route - pseudo-wire route

bgp-attribute-list (R/W)

contains a list of bgp attributes of the following BGP attributes

- \* bgp-origin - origin of BGP route (IGP, EGP or incomplete)
- \* bgp-aspath structure which contains use count (usascount), reference count (ulrefcount), AS (in string form), length of as sequence (usascountfrcmp), total number of as in path count (usastotalcount), length of 4-byte AS path tlvs (usaspathlen), and string with 4-byte AS path value (as4pathvalue).
- \* bgp-nexthop - BGP next hop

- \* bgp-med - Multi-Exit Discriminator
- \* bgp-localpref - Local Preference for route
- \* bgp-atomic-aggregation - ID for aggregator
- \* bgp-aggregator -IPv4 address (4 bytes) (ulipaddress), AS number (4 byte) (ulasnumber)
- \* bgp-community attribute (bgp-comm-attr) which contains: size of community attribute (uscommsize), reference count (ulrefcomm), and value (string).
- \* bgp extended community which contains: custom-cost community (valid flag, insertion pint, community id, and cost), size of community attribute (usexcommsize), reference count (ulrefcount), and external community value (string)(extcommattr-value),
- \* bgp cluster list which contains: lengthof cluster lists (uscluslen), reference count (ulrefclus), list of values (string)(clusterlist-value)

#### bgp\_route\_creation (R/W)

provides the status of the creation of a BGP route requested by a I2RS Client to I2RS agent to be created in the BGP instance's local route. The status includes the following:

- \* not-i2rs - routes created by non-I2RS sources
- \* i2rs-client-create-bgp-route - client requests a bgp local route created
- \* i2rsagent-fails-bgp-route-create - I2rs agent fails to create bgp local rib route
- \* i2rsagent-bgp-route-created - I2rsagent created this bgp local rib route
- \* i2rsagent-rejects-bgp-route-create - I2rs agent policy rejects attempt
- \* i2rsagent-attempts-bgp-route-create - route create in process

#### bgp\_status\_info (R/W)

contains a set of BGP status that includes:

```

*  rt_rib_current_state
*  rt_rib_last_state
*  rt_rib_reject_reason
*  rt_not_preferred

```

### 3.4. BGP Peer

This section describes the BGP peer structure.

#### 3.4.1. Yang Top Level for BGP Peer

```

+--rw bgp-peer-list* [bgp-peer-name]
|   +--rw peer-session-address
|   |   +--rw local-ipv4-address?    inet:ipv4-prefix
|   |   +--rw remote-ipv4-address?   inet:ipv4-prefix
|   +--rw bgp-peer-name              string
|   +--ro bgp-peer-type?              enumeration
|   +--ro bgp-peer-create?            enumeration
|   +--rw bgp-policy-in
|   +--rw bgp-policy-out
|   +--ro peer-policy-in-pset
|   +--ro peer-policy-out-pset
|   +--rw peer-state-info
|   |   +--ro peer-current-state?     peer-state
|   |   +--ro peer-last-state?       peer-state
|   |   +--ro peer-down-reason
|   |   |   +--ro error-code?         enumeration
|   |   |   +--ro sub-error-code
|   |   |   |   +--ro (sub-error-code-type)?
|   |   |   |   |   +--:(head-error-sub-code)
|   |   |   |   |   |   +--ro head-error-sub-value?  enumeration
|   |   |   |   |   +--:(open-error-sub-code)
|   |   |   |   |   |   +--ro open-error-sub-value?  enumeration
|   |   |   |   |   +--:(update-error-sub-code)
|   |   |   |   |   |   +--ro update-error-sub-value? enumeration
|   |   |   |   |   +--:(route-refresh-error-sub-code)
|   |   |   |   |   |   +--ro route-refresh-error-sub-value?
|   |   |   |   |   |   enumeration
|   |   +--ro peer-received-update-cnt? uint64
|   |   +--ro peer-transmit-update-cnt? uint64
|   |   +--ro peer-received-route-cnt?  uint64
|   |   +--ro peer-send-route-cnt?      uint64
|   |   +--rw max-prefix-rcv-limit?     uint64
|   |   +--rw max-prefix-xmt-limit?     uint64

```

```

|   +--ro peer-prefix-high?          uint64
|   +--ro peer-prefix-low?          uint64
|   +--ro peer-prefix-ave?          uint64
|   +--ro peer-prefix-time?         uint64
|   +--ro peer-prefix-max?          uint64
|   +--ro publish-version?          uint64
+--rw bgp-rib-in
  +--rw bgp-rib-in-list* [ipv4-route ipv4-prefix-length]
    +--rw ipv4-route                inet:ipv4-prefix
    +--rw ipv4-prefix-length        uint8
    +--rw bgp-route-type?           enumeration
    +--rw bgp-attribute-list
      ....
    +--ro bgp-route-create?         enumeration
    +--ro bgp-rt-state-info
      ....
+--rw bgp-rib-out
  +--rw bgp-rib-out-list* [ipv4-route ipv4-prefix-length]
    +--rw ipv4-route                inet:ipv4-prefix
    +--rw ipv4-prefix-length        uint8
    +--rw bgp-route-type?           enumeration
    +--rw bgp-attribute-list
      ....
    +--ro bgp-route-create?         enumeration
    +--ro bgp-rt-state-info
      ....

```

### 3.4.2. Elements in BGP peer Structure

The following are the elements in the BGP\_Peer structure

BGP\_PEER\_NAME (R/W)

This entry uniquely names a BGP Peer. If multiple TCP connections are utilized between two BGP Peers, then the BGP Peer uniquely identifies the Peering. An I2RS Write to a BGP\_PEER\_NAME changes the BGP\_PEER\_NAME For the duration of the I2RS client-agent association for the run-time BGP, but it does not change the configured BGP Peer name.

Peer\_Session\_Address (R/W)

provides local peer address information and remote peer address information. Please see the description below for the details of the BGP session address. An I2RS read of this information pulls this information pulls up the the BGP Peer data in ephemeral store. The I2RS write to this causes the peer to attempt to set-

up a TCP session either initial TCP session or a second TCP session in with multiple TCP support

BGP\_PEER\_TYPE (R/W)

contains the status on the type of BGP such as EBGp, IBGP, RR, RR-Client, PE, or CE. or Vendor specific types. (I2RS Note: Does this match normal configuration expectation or should this be an I2RS special variable)

BGP\_Peer\_create(R/W)

provides the I2RS creation status for this peer as: not-created by I2RS, requested by client but not processed, or an I2RS agent's response to the request: (agent reject peer creation, agent attempting peer creation, Agent failed peer creation, and Agent created BGP Peer).

Peer\_policy\_in (R/W)

- This allows reference of the bgp inbound policy by policy set name. Policy sets are defined by [RFC3460], and refers to the name of the policy set. This allows I2RS to read/write the policy set name, and let the I2RS agent link in the appropriate policy set.

Peer\_policy\_out (R/W)

- This allows reference of the outbound policy by policy set names, and let the I2RS agent load an existing policy set.

Peer\_policy\_in\_pset (R/W)

Full policy set loaded for Peer\_policy\_in\_pset.

Peer\_policy\_in\_pset (R/W)

Full policy set loaded for Peer\_policy\_in\_pset.

Peer state info (ro)

includes the following information:

- \* Peer\_current\_state (R) the the current state of the BGP Peer in the BGP state machine.
- \* Peer\_last\_state (R)

- \* Peer\_down\_reason (R) which includes the reason and the sub-error codes for bgp header errors, bgp open errors, bgp update errors, and bgp route refresh errors
- \* peer-received-update-cnt (ro) - received
- \* peer-transmit-update-cnt (ro)
- \* peer-received-route-cnt (ro)
- \* peer-send-route-cnt (ro)
- \* peer-max-prefix-rcv-limit (rw)
- \* peer-max-prefix-xmt-limit(rw)
- \* peer-prefix-high (ro)
- \* peer-prefix-low (ro)
- \* peer-prefix-ave (ro)
- \* peer-prefix-time (ro)
- \* peer-prefix-max (ro)
- \* publish-version (ro)

#### bgp\_rib\_in

Route received (AdjRIBIn) in tree expressed in yang as a list of routes index by IPv4-route and Ipv4-prefix-length. Each route contains the following:

- \* bgp-route-type (rw) - type of route (described above)
- \* bgp-attribute-list (rw) - described above)
- \* bgp-route-create (ro) - source config or I2RS create status (described above)
- \* bgp-route-state (ro) - state of route (described above)

#### bgp\_rib\_out (R/W)

- Route output (AdjRIBOut) in tree expressed in yang as a list of routes index by IPv4-route and Ipv4-prefix-length. Each route contains the following:



- \* `bgp-route-type` (rw) - type of route (described above)
- \* `bgp-attribute-list` (rw) - described above)
- \* `bgp-route-create` (ro) - source config or I2RS create status (described above)
- \* `bgp-route-state` (ro) - state of route (described above)

#### 3.4.3. Reading/Writing Policy in I2RS IM model for BGP Peer

The I2RS Agent's peer inbound and outbound policies are logically considered to be stored in policy sets that logically are defined in [RFC3460] but have the additions to be appropriate for I2RS basic network policy as described [I-D.hares-i2rs-bnp-info-model]. Policy groups uniquely identified by a Policy name. Policy sets are described either by name so that an on-board policy set is associated with the I2RS control BGP peer, or by full policy sets (-pset). This section describes why the choice to allow both name for pre-loaded policy sets and full policy set description.

Some I2RS clients will have long-term connections to I2RS Agents associated with a BGP routing function, and may choose to transfer the BGP Policy set or a set of the BGP Policy Components to the client's local memory. In this case, the I2RS Client could request the I2RS Agent to simply transfer the BGP Policy Sets name used by for the BGP Peers to minimize travel along with an indication of the last time the policy was updated.

The policy variables `Peer_policy_in` and `Peer_policy_out` allow both read and write of the Policy set names. The read of the `Peer_policy_in` and `Peer_policy_out` will provide simply the policy-set name. The write of the `Peer_policy_in` in the I2RS agent will need to validate the policy set name. After the policy set name is validated, the policy set link is placed in `Peer_policy_in_pset` and the BGP routing function updated with the policy set lists for the BGP Peer.

Similarly, if a write occurs to I2RS Agent's `Peer_policy_out`, the I2RS agent validates the policy-set name, replaces `Peer_policy_out_pset` with the link to the policy-set identified by the Policy-set name.

Reading either `Peer_policy_in_pset` will result in the I2RS Agent passing the policy linked to by the `Peer_policy_in` name (and linked to by `Peer_policy_in_pset`) to the I2RS Client in policy set form. Similarly, reading the `Peer_policy_out_pset` will retrieve the policy

set pointed to by the Peer\_policy\_out name, and transfer the data in policy set form.

### 3.5. Additional variable used within all RIBS

The following additional variables are defined in the additional routes types.

- o route-distinguisher (R/W) - defined as route-distinguisher-def
- o mpls-label (R/W) - defined as 24 bit mpls labels
- o EVPN\_AD\_ROUTE (R/W) contains the following
  - \* route-type - EVPN route type
  - \* route-distinguisher - EVPN route distringuisher
  - \* esi - string
  - \* eth-tag-id - 32 bit ethernet tag
- o evpn-mac-route contains the following
  - \* route-type (r/w) - evpn-route-type
  - \* route-distinguisher (r/w)
  - \* esi (r/w) - string
  - \* eth-tag-id (r/w) 32 bit ethernet tag
  - \* mac-length (r/w) - length mac address (uint8)
  - \* mac-address (r/w)
  - \* ip-address-length (r/w) (uint8)
  - \* ip-address (r/w)
  - \* label 1 (r/w) (uint32)
  - \* label 2 (r/2) (uint32)
- o evpn-mcast-tree-route contains the following
  - \* route-type (r/w)

- \* route-distinguisher (r/w)
- \* eth-tag-id (r/w) ethernet tag id
- \* ip address-length (r/w) uint32
- \* origin-router-ip-address (r/w)
- o evpn-eth-segment-route contains the following:
  - \* route-type (r/w) - evpn-route-type
  - \* route-distinguisher (r/w)
  - \* esi (r/w) - string ethernet segment id
  - \* ip address length(r/w) uint8
  - \* origin-router-ip-address (r/w)
  - \* FLOW\_route (R/W) - to be completed
  - \* MDT route (R/W) - to be completed

#### 4. BGP Yang top-level description

Below is the full BGP Yang top-level description

```
odule: bgp-protocol
  +--rw bgp-protocol
    +--rw router-id?                inet:ip-address
    +--rw as-number?                uint32
    +--ro protocol-status?          enumeration
    +--rw shutdown-protocol?        boolean
    +--ro bgp-role?                 enumeration
    +--rw bgp-ipv4-uni-instance
      | +--rw bgp-instance-name?    string
      | +--rw bgp-instance-create?  enumeration
      | +--rw bgp-instance-type?    enumeration
      | +--rw bgp-vendor-type?      enumeration
      | +--rw afi?                  afi-def
      | +--rw safi?                 safi-def
      | +--rw bgp-local-rib
      | | +--rw bgp-route-list* [ipv4-route ipv4-prefix-length]
      | | | +--rw ipv4-route        inet:ipv4-prefix
      | | | +--rw ipv4-prefix-length uint8
      | | | +--rw bgp-route-type?    enumeration
      | | | +--rw route-admin-distance uint16
```

```

+--rw bgp-attribute-list
|   +--rw bgp-origin?          enumeration
|   +--rw bgp-aspath
|   |   +--rw usascount?        uint16
|   |   +--rw ulrefcount?       uint16
|   |   +--rw asstring?         string
|   |   +--ro usascountfrcomp?  uint16
|   |   +--rw usastotalcount?   uint16
|   |   +--rw usas4pathlen?     uint16
|   |   +--rw as4pathvalue?     string
|   +--rw bgp-nexthop?         inet:ip-address
|   +--rw bgp-med?             uint32
|   +--rw bgp-localpref?       uint32
|   +--rw bgp-atomic-aggregate? uint32
|   +--rw bgp-aggregator
|   |   +--rw ulipaddress?      uint32
|   |   +--rw ulasnumber?      uint32
|   +--rw bgp-commattr
|   |   +--rw uscommsize?       uint16
|   |   +--rw ulrefcomm?       uint32
|   |   +--rw commattr-value?  string
|   +--rw bgp-extcommattr
|   |   +--rw
|   |   |   +--rw valid          boolean
|   |   |   +--rw insertion-point uint32
|   |   |   +--rw community-id   uint8
|   |   |   +--rw cost-id        uint32
|   |   +--rw usextcommsize?     uint16
|   |   +--rw ulrefcount?       uint32
|   |   +--rw extcommattr-value? string
|   +--rw bgp-clusterlist
|   |   +--rw uscluslen?        uint16
|   |   +--rw ulrefclus?       uint32
|   |   +--rw clusterlist-value? string
|   +--rw bgp-originator-id?   uint32
+--ro bgp-route-create?        enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state?    rib-state-def
|   +--ro rib-last-state?      rib-state-def
|   +--ro rib-rejected-reason?  enumeration
|   +--ro not-preferred-reason? enumeration
+--rw bgp-peer-list* [bgp-peer-name]
|   +--rw peer-session-address
|   |   +--rw local-ipv4-address? inet:ipv4-prefix
|   |   +--rw remote-ipv4-address? inet:ipv4-prefix
|   +--rw bgp-peer-name        string
|   +--ro bgp-peer-type?       enumeration
|   +--ro bgp-peer-create?     enumeration

```

```

+--rw bgp-policy-in
+--rw bgp-policy-out
+--ro peer-policy-in-pset
+--ro peer-policy-out-pset
+--rw peer-state-info
|   +--ro peer-current-state?      peer-state
|   +--ro peer-last-state?        peer-state
|   +--ro peer-down-reason
|   |   +--ro error-code?          enumeration
|   |   +--ro sub-error-code
|   |   |   +--ro (sub-error-code-type)?
|   |   |   |   +--:(head-error-sub-code)
|   |   |   |   |   +--ro head-error-sub-value? enumeration
|   |   |   |   |   +--:(open-error-sub-code)
|   |   |   |   |   |   +--ro open-error-sub-value? enumeration
|   |   |   |   |   |   +--:(update-error-sub-code)
|   |   |   |   |   |   |   +--ro update-error-sub-value? enumeration
|   |   |   |   |   |   |   +--:(route-refresh-error-sub-code)
|   |   |   |   |   |   |   |   +--ro route-refresh-error-sub-value?
|   |   |   |   |   |   |   |   enumeration
+--ro peer-received-update-cnt?    uint64
+--ro peer-transmit-update-cnt?    uint64
+--ro peer-received-route-cnt?     uint64
+--ro peer-send-route-cnt?         uint64
+--rw max-prefix-rcv-limit?        uint64
+--rw max-prefix-xmt-limit?        uint64
+--ro peer-prefix-high?            uint64
+--ro peer-prefix-low?             uint64
+--ro peer-prefix-ave?             uint64
+--ro peer-prefix-time?            uint64
+--ro peer-prefix-max?             uint64
+--ro publish-version?             uint64
+--rw bgp-rib-in
|   +--rw bgp-rib-in-list* [ipv4-route ipv4-prefix-length]
|   |   +--rw ipv4-route            inet:ipv4-prefix
|   |   +--rw ipv4-prefix-length    uint8
|   |   +--rw bgp-route-type?       enumeration
|   |   +--rw route-admin-distance  uint16
|   |   +--rw bgp-attribute-list
|   |   |   +--rw bgp-origin?        enumeration
|   |   |   +--rw bgp-aspath
|   |   |   |   +--rw usascount?      uint16
|   |   |   |   +--rw ulrefcount?     uint16
|   |   |   |   +--rw asstring?       string
|   |   |   |   +--ro usascountfrcomp? uint16
|   |   |   |   +--rw usastotalcount? uint16
|   |   |   |   +--rw usas4pathlen?   uint16
|   |   |   |   +--rw as4pathvalue?   string

```

```

|--rw bgp-nexthop?                inet:ip-address
+--rw bgp-med?                    uint32
+--rw bgp-localpref?              uint32
+--rw bgp-atomic-aggregate?       uint32
+--rw bgp-aggregator
|   +--rw ulipaddress?            uint32
|   +--rw ulasnumber?            uint32
+--rw bgp-commattr
|   +--rw uscommssize?            uint16
|   +--rw ulrefcomm?             uint32
|   +--rw commattr-value?        string
+--rw bgp-extcommattr
|   +--rw custom-community
|   |   +--rw valid              boolean
|   |   +--rw insertion-point    uint32
|   |   +--rw community-id      uint8
|   |   +--rw cost-id           uint32
|   +--rw usextcommssize?        uint16
|   +--rw ulrefcount?            uint32
|   +--rw extcommattr-value?     string
+--rw bgp-clusterlist
|   +--rw uscluslen?             uint16
|   +--rw ulrefclus?            uint32
|   +--rw clusterlist-value?    string
+--rw bgp-originator-id?         uint32
+--ro bgp-route-create?          enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state?      rib-state-def
|   +--ro rib-last-state?        rib-state-def
|   +--ro rib-rejected-reason?    enumeration
|   +--ro not-preferred-reason?   enumeration
+--rw bgp-rib-out
+--rw bgp-rib-out-list* [ipv4-route ipv4-prefix-length]
|   +--rw ipv4-route              inet:ipv4-prefix
|   +--rw ipv4-prefix-length      uint8
|   +--rw bgp-route-type?         enumeration
|   +--rw route-admin-distance    uint16
|   +--rw bgp-attribute-list
|   |   +--rw bgp-orig?           enumeration
|   |   +--rw bgp-aspath
|   |   |   +--rw usascount?      uint16
|   |   |   +--rw ulrefcount?     uint16
|   |   |   +--rw asstring?       string
|   |   |   +--ro usascountfrcomp? uint16
|   |   |   +--rw usastotalcount? uint16
|   |   |   +--rw usas4pathlen?   uint16
|   |   |   +--rw as4pathvalue?   string
|   +--rw bgp-nexthop?          inet:ip-address

```

```

+---rw bgp-med?                                uint32
+---rw bgp-localpref?                          uint32
+---rw bgp-atomic-aggregate?                  uint32
+---rw bgp-aggregator
|   +---rw ulipaddress?                        uint32
|   +---rw ulasnumber?                        uint32
+---rw bgp-commattr
|   +---rw uscommssize?                        uint16
|   +---rw ulrefcomm?                         uint32
|   +---rw commattr-value?                    string
+---rw bgp-extcommattr
|   +---rw custom-community
|   |   +---rw valid                          boolean
|   |   +---rw insertion-point                uint32
|   |   +---rw community-id                  uint8
|   |   +---rw cost-id                       uint32
|   +---rw usextcommsize?                     uint16
|   +---rw ulrefcount?                       uint32
|   +---rw extcommattr-value?                 string
+---rw bgp-clusterlist
|   +---rw uscluslen?                         uint16
|   +---rw ulrefclus?                        uint32
|   +---rw clusterlist-value?                 string
+---rw bgp-originator-id?                     uint32
+---ro bgp-route-create?                       enumeration
+---ro bgp-rt-state-info
|   +---ro rib-current-state?                 rib-state-def
|   +---ro rib-last-state?                   rib-state-def
|   +---ro rib-rejected-reason?              enumeration
|   +---ro not-preferred-reason?             enumeration
+---rw bgp-labeled-ipv4-instance
|   +---rw bgp-instance-name?                string
|   +---rw bgp-instance-create?              enumeration
|   +---rw bgp-instance-type?                enumeration
|   +---rw bgp-vendor-type?                  enumeration
|   +---rw afi?                              afi-def
|   +---rw safi?                              safi-def
|   +---rw bgp-local-rib
|   |   +---rw bgp-route-list*
|   |   |   [ipv4-route ipv4-prefix-length mpls-label]
|   |   |   +---rw ipv4-route                inet:ipv4-prefix
|   |   |   +---rw ipv4-prefix-length        uint8
|   |   |   +---rw mpls-label                mpls-label-def
|   |   |   +---rw bgp-route-type?           enumeration
|   |   |   +---rw route-admin-distance     uint16
|   |   +---rw bgp-attribute-list
|   |   |   +---rw bgp-origin?                enumeration
|   |   |   +---rw bgp-aspath

```

```

+---rw usascount?                uint16
+---rw ulrefcount?               uint16
+---rw asstring?                 string
+---ro usascountfrcomp?         uint16
+---rw usastotalcount?          uint16
+---rw usas4pathlen?            uint16
+---rw as4pathvalue?             string
+---rw bgp-nexthop?              inet:ip-address
+---rw bgp-med?                  uint32
+---rw bgp-localpref?            uint32
+---rw bgp-atomic-aggregate?     uint32
+---rw bgp-aggregator
|   +---rw ulipaddress?          uint32
|   +---rw ulasnumber?           uint32
+---rw bgp-commattr
|   +---rw uscommssize?           uint16
|   +---rw ulrefcomm?            uint32
|   +---rw commattr-value?       string
+---rw bgp-extcommattr
|   +---rw custom-community
|       +---rw valid              boolean
|       +---rw insertion-point    uint32
|       +---rw community-id       uint8
|       +---rw cost-id            uint32
|   +---rw usextcommssize?        uint16
|   +---rw ulrefcount?            uint32
|   +---rw extcommattr-value?     string
+---rw bgp-clusterlist
|   +---rw uscluslen?             uint16
|   +---rw ulrefclus?             uint32
|   +---rw clusterlist-value?     string
+---rw bgp-originator-id?        uint32
+---ro bgp-route-create?          enumeration
+---ro bgp-rt-state-info
|   +---ro rib-current-state?     rib-state-def
|   +---ro rib-last-state?        rib-state-def
|   +---ro rib-rejected-reason?    enumeration
|   +---ro not-preferred-reason?    enumeration
+---rw bgp-peer-list* [bgp-peer-name]
+---rw peer-session-address
|   +---rw local-ipv4-address?     inet:ipv4-prefix
|   +---rw remote-ipv4-address?    inet:ipv4-prefix
+---rw bgp-peer-name              string
+---ro bgp-peer-type?             enumeration
+---ro bgp-peer-create?           enumeration
+---rw bgp-policy-in
+---rw bgp-policy-out
+---ro peer-policy-in-pset

```



```

+--ro peer-policy-out-pset
+--rw peer-state-info
|   +--ro peer-current-state?          peer-state
|   +--ro peer-last-state?             peer-state
|   +--ro peer-down-reason
|   |   +--ro error-code?              enumeration
|   |   +--ro sub-error-code
|   |   |   +--ro (sub-error-code-type)?
|   |   |   |   +--:(head-error-sub-code)
|   |   |   |   |   +--ro head-error-sub-value? enumeration
|   |   |   |   +--:(open-error-sub-code)
|   |   |   |   |   +--ro open-error-sub-value? enumeration
|   |   |   |   +--:(update-error-sub-code)
|   |   |   |   |   +--ro update-error-sub-value? enumeration
|   |   |   |   +--:(route-refresh-error-sub-code)
|   |   |   |   +--ro route-refresh-error-sub-value?
|   |   |   |   enumeration
|   +--ro peer-received-update-cnt?    uint64
|   +--ro peer-transmit-update-cnt?    uint64
|   +--ro peer-received-route-cnt?     uint64
|   +--ro peer-send-route-cnt?         uint64
|   +--rw max-prefix-rcv-limit?        uint64
|   +--rw max-prefix-xmt-limit?        uint64
|   +--ro peer-prefix-high?            uint64
|   +--ro peer-prefix-low?             uint64
|   +--ro peer-perfix-ave?             uint64
|   +--ro peer-prefix-time?            uint64
|   +--ro peer-prefix-max?             uint64
|   +--ro publish-version?             uint64
+--rw bgp-rib-in
|   +--rw bgp-rib-in-list*
|   |   [ipv4-route ipv4-prefix-length mpls-label]
|   |   +--rw ipv4-route               inet:ipv4-prefix
|   |   +--rw ipv4-prefix-length       uint8
|   |   +--rw mpls-label                mpls-label-def
|   |   +--rw bgp-route-type?          enumeration
|   |   +--rw route-admin-distance     uint16
|   |   +--rw bgp-attribute-list
|   |   |   +--rw bgp-origin?          enumeration
|   |   |   +--rw bgp-aspath
|   |   |   |   +--rw usascount?        uint16
|   |   |   |   +--rw ulrefcount?       uint16
|   |   |   |   +--rw asstring?         string
|   |   |   |   +--ro usascountfrcomp?  uint16
|   |   |   |   +--rw usastotalcount?   uint16
|   |   |   |   +--rw usas4pathlen?     uint16
|   |   |   |   +--rw as4pathvalue?     string
|   |   +--rw bgp-nexthop?             inet:ip-address

```

```

+---rw bgp-med?                               uint32
+---rw bgp-localpref?                          uint32
+---rw bgp-atomic-aggregate?                   uint32
+---rw bgp-aggregator
|   +---rw ulipaddress?                        uint32
|   +---rw ulasnumber?                        uint32
+---rw bgp-commattr
|   +---rw uscommsize?                         uint16
|   +---rw ulrefcomm?                         uint32
|   +---rw commattr-value?                    string
+---rw bgp-extcommattr
|   +---rw custom-community
|   |   +---rw valid                          boolean
|   |   +---rw insertion-point               uint32
|   |   +---rw community-id                  uint8
|   |   +---rw cost-id                       uint32
|   +---rw usextcommsize?                     uint16
|   +---rw ulrefcount?                       uint32
|   +---rw extcommattr-value?                 string
+---rw bgp-clusterlist
|   +---rw uscluslen?                         uint16
|   +---rw ulrefclus?                        uint32
|   +---rw clusterlist-value?                string
+---rw bgp-originator-id?                     uint32
+---ro bgp-route-create?                       enumeration
+---ro bgp-rt-state-info
|   +---ro rib-current-state?                 rib-state-def
|   +---ro rib-last-state?                   rib-state-def
|   +---ro rib-rejected-reason?              enumeration
|   +---ro not-preferred-reason?             enumeration
+---rw bgp-rib-out
+---rw bgp-rib-out-list*
|   [ipv4-route ipv4-prefix-length mpls-label]
+---rw ipv4-route                             inet:ipv4-prefix
+---rw ipv4-prefix-length                     uint8
+---rw mpls-label                             mpls-label-def
+---rw bgp-route-type?                       enumeration
+---rw route-admin-distance                   uint16
+---rw bgp-attribute-list
|   +---rw bgp-origin?                       enumeration
|   +---rw bgp-aspath
|   |   +---rw usascount?                     uint16
|   |   +---rw ulrefcount?                   uint16
|   |   +---rw asstring?                     string
|   |   +---ro usascountfrcomp?              uint16
|   |   +---rw usastotalcount?               uint16
|   |   +---rw usas4pathlen?                 uint16
|   |   +---rw as4pathvalue?                 string

```

```

|--rw bgp-nexthop?                inet:ip-address
+--rw bgp-med?                     uint32
+--rw bgp-localpref?               uint32
+--rw bgp-atomic-aggregate?       uint32
+--rw bgp-aggregator
|   +--rw ulipaddress?             uint32
|   +--rw ulasnumber?             uint32
+--rw bgp-commattr
|   +--rw uscommssize?             uint16
|   +--rw ulrefcomm?              uint32
|   +--rw commattr-value?         string
+--rw bgp-extcommattr
|   +--rw custom-community
|   |   +--rw valid                boolean
|   |   +--rw insertion-point     uint32
|   |   +--rw community-id       uint8
|   |   +--rw cost-id             uint32
|   +--rw usextcommssize?         uint16
|   +--rw ulrefcount?             uint32
|   +--rw extcommattr-value?     string
+--rw bgp-clusterlist
|   +--rw uscluslen?              uint16
|   +--rw ulrefclus?             uint32
|   +--rw clusterlist-value?     string
+--rw bgp-originator-id?          uint32
+--ro bgp-route-create?            enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state?      rib-state-def
|   +--ro rib-last-state?        rib-state-def
|   +--ro rib-rejected-reason?    enumeration
|   +--ro not-preferred-reason?  enumeration
+--rw bgp-ipv4-multi-instance
|   +--rw bgp-instance-name?      string
|   +--rw bgp-instance-create?    enumeration
|   +--rw bgp-instance-type?      enumeration
|   +--rw bgp-vendor-type?        enumeration
|   +--rw afi?                    afi-def
|   +--rw safi?                    safi-def
+--rw bgp-local-rib
|   +--rw bgp-route-list* [ipv4-route ipv4-prefix-length]
|   |   +--rw ipv4-route          inet:ipv4-prefix
|   |   +--rw ipv4-prefix-length  uint8
|   |   +--rw bgp-route-type?     enumeration
|   |   +--rw route-admin-distance uint16
|   +--rw bgp-attribute-list
|   |   +--rw bgp-origin?          enumeration
|   |   +--rw bgp-aspath
|   |   |   +--rw usascount?       uint16

```

```

| | +---rw ulrefcount?                uint16
| | +---rw asstring?                  string
| | +---ro usascountfrcomp?          uint16
| | +---rw usastotalcount?           uint16
| | +---rw usas4pathlen?             uint16
| | +---rw as4pathvalue?             string
+---rw bgp-nexthop?                  inet:ip-address
+---rw bgp-med?                      uint32
+---rw bgp-localpref?               uint32
+---rw bgp-atomic-aggregate?        uint32
+---rw bgp-aggregator
|   +---rw ulipaddress?              uint32
|   +---rw ulasnumber?               uint32
+---rw bgp-commattr
|   +---rw uscommsize?                uint16
|   +---rw ulrefcomm?                uint32
|   +---rw commattr-value?            string
+---rw bgp-extcommattr
|   +---rw custom-community
|       +---rw valid                  boolean
|       +---rw community-id           uint8
|       +---rw insertion-point         uint32
|       +---rw cost-id                uint32
|   +---rw usextcommsize?             uint16
|   +---rw ulrefcount?                uint32
|   +---rw extcommattr-value?         string
+---rw bgp-clusterlist
|   +---rw uscluslen?                 uint16
|   +---rw ulrefclust?                uint32
|   +---rw clusterlist-value?         string
+---rw bgp-originator-id?            uint32
+---ro bgp-route-create?              enumeration
+---ro bgp-rt-state-info
+---ro rib-current-state?             rib-state-def
+---ro rib-last-state?               rib-state-def
+---ro rib-rejected-reason?          enumeration
+---ro not-preferred-reason?         enumeration
+---rw bgp-peer-list* [bgp-peer-name]
+---rw peer-session-address
|   +---rw local-ipv4-address?        inet:ipv4-prefix
|   +---rw remote-ipv4-address?       inet:ipv4-prefix
+---rw bgp-peer-name                  string
+---ro bgp-peer-type?                 enumeration
+---ro bgp-peer-create?               enumeration
+---rw bgp-policy-in
+---rw bgp-policy-out
+---ro peer-policy-in-pset
+---ro peer-policy-out-pset

```

```

+--rw peer-state-info
|   +--ro peer-current-state?          peer-state
|   +--ro peer-last-state?             peer-state
|   +--ro peer-down-reason
|   |   +--ro error-code?              enumeration
|   |   +--ro sub-error-code
|   |   |   +--ro (sub-error-code-type)?
|   |   |   |   +--:(head-error-sub-code)
|   |   |   |   |   +--ro head-error-sub-value? enumeration
|   |   |   |   +--:(open-error-sub-code)
|   |   |   |   |   +--ro open-error-sub-value? enumeration
|   |   |   |   +--:(update-error-sub-code)
|   |   |   |   |   +--ro update-error-sub-value?
|   |   |   |   |   enumeration
|   |   |   +--:(route-refresh-error-sub-code)
|   |   |   |   +--ro route-refresh-error-sub-value?
|   |   |   |   |   enumeration
|   |   +--ro peer-received-update-cnt? uint64
|   |   +--ro peer-transmit-update-cnt? uint64
|   |   +--ro peer-received-route-cnt?  uint64
|   |   +--ro peer-send-route-cnt?      uint64
|   |   +--rw max-prefix-rcv-limit?     uint64
|   |   +--rw max-prefix-xmt-limit?     uint64
|   |   +--ro peer-prefix-high?         uint64
|   |   +--ro peer-prefix-low?          uint64
|   |   +--ro peer-perfix-ave?          uint64
|   |   +--ro peer-prefix-time?         uint64
|   |   +--ro peer-prefix-max?          uint64
|   |   +--ro publish-version?          uint64
|   +--rw bgp-rib-in
|   |   +--rw bgp-rib-in-list* [ipv4-route ipv4-prefix-length]
|   |   |   +--rw ipv4-route            inet:ipv4-prefix
|   |   |   +--rw ipv4-prefix-length    uint8
|   |   |   +--rw bgp-route-type?       enumeration
|   |   |   +--rw route-admin-distance uint16
|   |   |   +--rw bgp-attribute-list
|   |   |   |   +--rw bgp-origin?        enumeration
|   |   |   |   +--rw bgp-aspath
|   |   |   |   |   +--rw usascount?      uint16
|   |   |   |   |   +--rw ulrefcount?     uint16
|   |   |   |   |   +--rw asstring?       string
|   |   |   |   |   +--ro usascountfrcomp? uint16
|   |   |   |   |   +--rw usastotalcount? uint16
|   |   |   |   |   +--rw usas4pathlen?   uint16
|   |   |   |   |   +--rw as4pathvalue?   string
|   |   |   +--rw bgp-nexthop?          inet:ip-address
|   |   |   +--rw bgp-med?               uint32
|   |   |   +--rw bgp-localpref?        uint32

```

```

|--rw bgp-atomic-aggregate?      uint32
+--rw bgp-aggregator
|   +--rw ulipaddress?           uint32
|   +--rw ulasnumber?           uint32
+--rw bgp-commattr
|   +--rw uscommssize?           uint16
|   +--rw ulrefcomm?            uint32
|   +--rw commattr-value?       string
+--rw bgp-extcommattr
|   +--rw custom-community
|   |   +--rw valid              boolean
|   |   +--rw insertion-point   uint32
|   |   +--rw community-id      uint8
|   |   +--rw cost-id           uint32
|   +--rw usextcommssize?       uint16
|   +--rw ulrefcount?           uint32
|   +--rw extcommattr-value?    string
+--rw bgp-clusterlist
|   +--rw uscluslen?            uint16
|   +--rw ulrefclus?           uint32
|   +--rw clusterlist-value?    string
+--rw bgp-originator-id?        uint32
+--ro bgp-route-create?          enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state?     rib-state-def
|   +--ro rib-last-state?       rib-state-def
|   +--ro rib-rejected-reason?   enumeration
|   +--ro not-preferred-reason?  enumeration
+--rw bgp-rib-out
+--rw bgp-rib-out-list* [ipv4-route ipv4-prefix-length]
|   +--rw ipv4-route             inet:ipv4-prefix
|   +--rw ipv4-prefix-length     uint8
|   +--rw bgp-route-type?       enumeration
|   +--rw route-admin-distance  uint16
+--rw bgp-attribute-list
|   +--rw bgp-origin?           enumeration
|   +--rw bgp-aspath
|   |   +--rw usascount?        uint16
|   |   +--rw ulrefcount?       uint16
|   |   +--rw asstring?         string
|   |   +--ro usascountfrcomp?   uint16
|   |   +--rw usastotalcount?    uint16
|   |   +--rw usas4pathlen?     uint16
|   |   +--rw as4pathvalue?     string
|   +--rw bgp-nexthop?         inet:ip-address
|   +--rw bgp-med?             uint32
|   +--rw bgp-localpref?       uint32
|   +--rw bgp-atomic-aggregate? uint32

```

```

| | | | +--rw bgp-aggregator
| | | | |   |--rw ulipaddress?      uint32
| | | | |   |--rw ulasnumber?     uint32
| | | | +--rw bgp-commattr
| | | | |   |--rw uscommsize?       uint16
| | | | |   |--rw ulrefcomm?       uint32
| | | | |   |--rw commattr-value?  string
| | | | +--rw bgp-extcommattr
| | | | |   |--rw custom-community
| | | | |   | | |--rw valid          boolean
| | | | |   | | |--rw insertion-point uint32
| | | | |   | | |--rw community-id  uint8
| | | | |   | | |--rw cost-id       uint32
| | | | |   |--rw usextcommsize?    uint16
| | | | |   |--rw ulrefcount?       uint32
| | | | |   |--rw extcommattr-value? string
| | | | +--rw bgp-clusterlist
| | | | |   |--rw uscluslen?         uint16
| | | | |   |--rw ulrefclust?       uint32
| | | | |   |--rw clusterlist-value? string
| | | | |   |--rw bgp-originator-id? uint32
| | | | +--ro bgp-route-create?     enumeration
| | | | +--ro bgp-rt-state-info
| | | | |   |--ro rib-current-state?  rib-state-def
| | | | |   |--ro rib-last-state?    rib-state-def
| | | | |   |--ro rib-rejected-reason? enumeration
| | | | |   |--ro not-preferred-reason? enumeration
+--rw bgp-vpnv4-instance
| | | | +--rw bgp-instance-name?    string
| | | | +--rw bgp-instance-create?  enumeration
| | | | +--rw bgp-instance-type?    enumeration
| | | | +--rw bgp-vendor-type?      enumeration
| | | | +--rw afi?                  afi-def
| | | | +--rw safi?                 safi-def
| | | | +--rw bgp-local-rib
| | | | |   +--rw bgp-route-list*
| | | | |   [ipv4-route ipv4-prefix-length route-distinguisher]
| | | | |   |   |--rw ipv4-route      inet:ipv4-prefix
| | | | |   |   |--rw ipv4-prefix-length uint8
| | | | |   |   |--rw mpls-label?     mpls-label-def
| | | | |   |   |--rw route-distinguisher route-distinguisher-def
| | | | |   |   |--rw bgp-route-type? enumeration
| | | | |   |   |--rw route-admin-distance uint16
| | | | |   |   |--rw bgp-attribute-list
| | | | |   |   |   |--rw bgp-origin? enumeration
| | | | |   |   |   |--rw bgp-aspath
| | | | |   |   |   |   |--rw usascount?      uint16
| | | | |   |   |   |   |--rw ulrefcount?     uint16

```

```

| | | | | +---rw asstring? string
| | | | | +---ro usascountfrcomp? uint16
| | | | | +---rw usastotalcount? uint16
| | | | | +---rw usas4pathlen? uint16
| | | | | +---rw as4pathvalue? string
| | | | +---rw bgp-nexthop? inet:ip-address
| | | | +---rw bgp-med? uint32
| | | | +---rw bgp-localpref? uint32
| | | | +---rw bgp-atomic-aggregate? uint32
| | | | +---rw bgp-aggregator
| | | | | +---rw ulipaddress? uint32
| | | | | +---rw ulasnumber? uint32
| | | | +---rw bgp-commattr
| | | | | +---rw uscommssize? uint16
| | | | | +---rw ulrefcomm? uint32
| | | | | +---rw commattr-value? string
| | | | +---rw bgp-extcommattr
| | | | | +---rw custom-community
| | | | | | +---rw valid boolean
| | | | | | +---rw insertion-point uint32
| | | | | | +---rw community-id uint8
| | | | | | +---rw cost-id uint32
| | | | | +---rw usextcommssize? uint16
| | | | | +---rw ulrefcount? uint32
| | | | | +---rw extcommattr-value? string
| | | | +---rw bgp-clusterlist
| | | | | +---rw uscluslen? uint16
| | | | | +---rw ulrefclust? uint32
| | | | | +---rw clusterlist-value? string
| | | | +---rw bgp-originator-id? uint32
+---ro bgp-route-create? enumeration
+---ro bgp-rt-state-info
| | | | +---ro rib-current-state? rib-state-def
| | | | +---ro rib-last-state? rib-state-def
| | | | +---ro rib-rejected-reason? enumeration
| | | | +---ro not-preferred-reason? enumeration
+---rw bgp-peer-list* [bgp-peer-name]
| | | | +---rw peer-session-address
| | | | | +---rw local-ipv4-address? inet:ipv4-prefix
| | | | | +---rw remote-ipv4-address? inet:ipv4-prefix
+---rw bgp-peer-name string
+---ro bgp-peer-type? enumeration
+---ro bgp-peer-create? enumeration
+---rw bgp-policy-in policy-group-name
+---rw bgp-policy-out policy-group-name
+---ro peer-policy-in-pset policy-set-name
+---ro peer-policy-out-pset policy-set-name
+---rw peer-state-info

```



```

+--ro peer-current-state?          peer-state
+--ro peer-last-state?             peer-state
+--ro peer-down-reason
|   +--ro error-code?              enumeration
|   +--ro sub-error-code
|       +--ro (sub-error-code-type)?
|           +--:(head-error-sub-code)
|               |   +--ro head-error-sub-value? enumeration
|           +--:(open-error-sub-code)
|               |   +--ro open-error-sub-value? enumeration
|           +--:(update-error-sub-code)
|               |   +--ro update-error-sub-value?
|                   enumeration
|           +--:(route-refresh-error-sub-code)
|               +--ro route-refresh-error-sub-value?
|                   enumeration
+--ro peer-received-update-cnt?    uint64
+--ro peer-transmit-update-cnt?    uint64
+--ro peer-received-route-cnt?    uint64
+--ro peer-send-route-cnt?        uint64
+--rw max-prefix-rcv-limit?       uint64
+--rw max-prefix-xmt-limit?       uint64
+--ro peer-prefix-high?           uint64
+--ro peer-prefix-low?            uint64
+--ro peer-perfix-ave?             uint64
+--ro peer-prefix-time?           uint64
+--ro peer-prefix-max?            uint64
+--ro publish-version?            uint64
+--rw bgp-rib-in
|   +--rw bgp-rib-in-list*
|       [ipv4-route ipv4-prefix-length route-distinguisher]
|           +--rw ipv4-route        inet:ipv4-prefix
|           +--rw ipv4-prefix-length uint8
|           +--rw mpls-label?       mpls-label-def
|           +--rw route-distinguisher route-distinguisher-def
|           +--rw bgp-route-type?   enumeration
|           +--rw route-admin-distance uint16
|           +--rw bgp-attribute-list
|               +--rw bgp-origin?   enumeration
|               +--rw bgp-aspath
|                   +--rw usascount? uint16
|                   +--rw ulrefcount? uint16
|                   +--rw asstring?  string
|                   +--ro usascountfrcomp? uint16
|                   +--rw usastotalcount? uint16
|                   +--rw usas4pathlen? uint16
|                   +--rw as4pathvalue? string
|           +--rw bgp-nexthop      inet:ip-address

```

```

+--rw bgp-med?                               uint32
+--rw bgp-localpref?                          uint32
+--rw bgp-atomic-aggregate?                  uint32
+--rw bgp-aggregator
|   +--rw ulipaddress?                        uint32
|   +--rw ulasnumber?                        uint32
+--rw bgp-commattr
|   +--rw uscommsize?                        uint16
|   +--rw ulrefcomm?                         uint32
|   +--rw commattr-value?                    string
+--rw bgp-extcommattr
|   +--rw custom-community
|   |   +--rw valid                          boolean
|   |   +--rw insertion-point                uint32
|   |   +--rw community-id                  uint8
|   |   +--rw cost-id                       uint32
|   +--rw usextcommsize?                     uint16
|   +--rw ulrefcount?                        uint32
|   +--rw extcommattr-value?                 string
+--rw bgp-clusterlist
|   +--rw uscluslen?                         uint16
|   +--rw ulrefclus?                         uint32
|   +--rw clusterlist-value?                string
+--rw bgp-originator-id?                     uint32
+--ro bgp-route-create?                       enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state?                 rib-state-def
|   +--ro rib-last-state?                    rib-state-def
|   +--ro rib-rejected-reason?               enumeration
|   +--ro not-preferred-reason?              enumeration
+--rw bgp-rib-out
+--rw bgp-rib-out-list*
|   [ipv4-route ipv4-prefix-length route-distinguisher]
|   +--rw ipv4-route                          inet:ipv4-prefix
|   +--rw ipv4-prefix-length                  uint8
|   +--rw mpls-label?                         mpls-label-def
|   +--rw route-distinguisher                 route-distinguisher-def
|   +--rw bgp-route-type?                     enumeration
|   +--rw route-admin-distance                uint16
|   +--rw bgp-attribute-list
|   |   +--rw bgp-origin?                     enumeration
|   |   +--rw bgp-aspath
|   |   |   +--rw usascount?                  uint16
|   |   |   +--rw ulrefcount?                 uint16
|   |   |   +--rw asstring?                   string
|   |   |   +--ro usascountfrcomp?            uint16
|   |   |   +--rw usastotalcount?             uint16
|   |   |   +--rw usas4pathlen?              uint16

```

```

+---rw as4pathvalue?      string
+---rw bgp-nexthop?      inet:ip-address
+---rw bgp-med?           uint32
+---rw bgp-localpref?     uint32
+---rw bgp-atomic-aggregate? uint32
+---rw bgp-aggregator
|   +---rw ulipaddress?    uint32
|   +---rw ulasnumber?     uint32
+---rw bgp-commattr
|   +---rw uscommsize?     uint16
|   +---rw ulrefcomm?      uint32
|   +---rw commattr-value? string
+---rw bgp-extcommattr
|   +---rw custom-community
|   |   +---rw valid       boolean
|   |   +---rw insertion-point uint32
|   |   +---rw community-id  uint8
|   |   +---rw cost-id      uint32
|   +---rw usextcommsize?   uint16
|   +---rw ulrefcount?      uint32
|   +---rw extcommattr-value? string
+---rw bgp-clusterlist
|   +---rw uscluslen?       uint16
|   +---rw ulrefclus?       uint32
|   +---rw clusterlist-value? string
+---rw bgp-originator-id?  uint32
+---ro bgp-route-create?    enumeration
+---ro bgp-rt-state-info
|   +---ro rib-current-state?    rib-state-def
|   +---ro rib-last-state?       rib-state-def
|   +---ro rib-rejected-reason?  enumeration
|   +---ro not-preferred-reason? enumeration
+---rw bgp-ipv4-l3vpn-instance-list* [bgp-l3vpn-instance-name]
+---rw bgp-l3vpn-instance-name      string
+---rw bgp-instance-name?            string
+---rw bgp-instance-create?          enumeration
+---rw bgp-instance-type?            enumeration
+---rw bgp-vendor-type?              enumeration
+---rw afi?                          afi-def
+---rw safi?                         safi-def
+---rw bgp-local-rib
|   +---rw bgp-route-list* [ipv4-route ipv4-prefix-length]
|   +---rw ipv4-route          inet:ipv4-prefix
|   +---rw ipv4-prefix-length  uint8
|   +---rw bgp-route-type?     enumeration
|   +---rw route-admin-distance uint16
|   +---rw bgp-attribute-list
|   |   +---rw bgp-origqn?      enumeration

```

```

+--rw bgp-aspath
|   +--rw usascount?          uint16
|   +--rw ulrefcount?        uint16
|   +--rw asstring?          string
|   +--ro usascountfrcomp?    uint16
|   +--rw usastotalcount?    uint16
|   +--rw usas4pathlen?      uint16
|   +--rw as4pathvalue?      string
+--rw bgp-nexthop?   inet:ip-address
+--rw bgp-med?       uint32
+--rw bgp-localpref? uint32
+--rw bgp-atomic-aggregate? uint32
+--rw bgp-aggregator
|   +--rw ulipaddress?   uint32
|   +--rw ulasnumber?   uint32
+--rw bgp-commattr
|   +--rw uscommsize?    uint16
|   +--rw ulrefcomm?    uint32
|   +--rw commattr-value? string
+--rw bgp-extcommattr
|   +--rw custom-community
|   |   +--rw valid          boolean
|   |   +--rw insertion-point uint32
|   |   +--rw community-id   uint8
|   |   +--rw cost-id        uint32
|   +--rw usextcommsize?    uint16
|   +--rw ulrefcount?       uint32
|   +--rw extcommattr-value? string
+--rw bgp-clusterlist
|   +--rw uscluslen?        uint16
|   +--rw ulrefclus?        uint32
|   +--rw clusterlist-value? string
+--rw bgp-originator-id?    uint32
+--ro bgp-route-create?     enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state?   rib-state-def
|   +--ro rib-last-state?     rib-state-def
|   +--ro rib-rejected-reason? enumeration
|   +--ro not-preferred-reason? enumeration
+--rw bgp-peer-list* [bgp-peer-name]
|   +--rw peer-session-address
|   |   +--rw local-ipv4-address?   inet:ipv4-prefix
|   |   +--rw remote-ipv4-address?  inet:ipv4-prefix
|   +--rw bgp-peer-name             string
|   +--ro bgp-peer-type?             enumeration
|   +--ro bgp-peer-create?           enumeration
+--rw bgp-policy-in
+--rw bgp-policy-out

```

```

+--ro peer-policy-in-pset
+--ro peer-policy-out-pset
+--rw peer-state-info
|   +--ro peer-current-state?          peer-state
|   +--ro peer-last-state?            peer-state
|   +--ro peer-down-reason
|   |   +--ro error-code?              enumeration
|   |   +--ro sub-error-code
|   |   |   +--ro (sub-error-code-type)?
|   |   |   |   +--:(head-error-sub-code)
|   |   |   |   |   +--ro head-error-sub-value? enumeration
|   |   |   |   +--:(open-error-sub-code)
|   |   |   |   |   +--ro open-error-sub-value? enumeration
|   |   |   |   +--:(update-error-sub-code)
|   |   |   |   |   +--ro update-error-sub-value?
|   |   |   |   |   enumeration
|   |   |   |   +--:(route-refresh-error-sub-code)
|   |   |   |   +--ro route-refresh-error-sub-value?
|   |   |   |   enumeration
|   |   +--ro peer-received-update-cnt? uint64
|   |   +--ro peer-transmit-update-cnt? uint64
|   |   +--ro peer-received-route-cnt?  uint64
|   |   +--ro peer-send-route-cnt?      uint64
|   |   +--rw max-prefix-rcv-limit?     uint64
|   |   +--rw max-prefix-xmt-limit?     uint64
|   |   +--ro peer-prefix-high?         uint64
|   |   +--ro peer-prefix-low?          uint64
|   |   +--ro peer-perfix-ave?          uint64
|   |   +--ro peer-prefix-time?         uint64
|   |   +--ro peer-prefix-max?          uint64
|   |   +--ro publish-version?          uint64
+--rw bgp-rib-in
|   +--rw bgp-rib-in-list*
|   |   [ipv4-route ipv4-prefix-length]
|   |   +--rw ipv4-route                inet:ipv4-prefix
|   |   +--rw ipv4-prefix-length         uint8
|   |   +--rw bgp-route-type?            enumeration
|   |   +--rw route-admin-distance       uint16
|   |   +--rw bgp-attribute-list
|   |   |   +--rw bgp-origin?            enumeration
|   |   |   +--rw bgp-aspath
|   |   |   |   +--rw usascount?          uint16
|   |   |   |   +--rw ulrefcount?         uint16
|   |   |   |   +--rw asstring?          string
|   |   |   |   +--ro usascountfrcomp?    uint16
|   |   |   |   +--rw usastotalcount?     uint16
|   |   |   |   +--rw usas4pathlen?      uint16
|   |   |   |   +--rw as4pathvalue?      string

```

```

|--rw bgp-nexthop?      inet:ip-address
+--rw bgp-med?          uint32
+--rw bgp-localpref?    uint32
+--rw bgp-atomic-aggregate?  uint32
+--rw bgp-aggregator
|   +--rw ulipaddress?    uint32
|   +--rw ulasnumber?    uint32
+--rw bgp-commattr
|   +--rw uscommssize?    uint16
|   +--rw ulrefcomm?     uint32
|   +--rw commattr-value? string
+--rw bgp-extcommattr
|   +--rw custom-community
|   |   +--rw valid      boolean
|   |   +--rw insertion-point uint32
|   |   +--rw community-id uint8
|   |   +--rw cost-id    uint32
|   +--rw usextcommssize? uint16
|   +--rw ulrefcount?    uint32
|   +--rw extcommattr-value? string
+--rw bgp-clusterlist
|   +--rw uscluslen?     uint16
|   +--rw ulrefclus?    uint32
|   +--rw clusterlist-value? string
+--rw bgp-originator-id? uint32
+--ro bgp-route-create?   enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state?  rib-state-def
|   +--ro rib-last-state?    rib-state-def
|   +--ro rib-rejected-reason? enumeration
|   +--ro not-preferred-reason? enumeration
+--rw bgp-rib-out
+--rw bgp-rib-out-list* [ipv4-route ipv4-prefix-length]
+--rw ipv4-route          inet:ipv4-prefix
+--rw ipv4-prefix-length  uint8
+--rw bgp-route-type?     enumeration
+--rw route-admin-distance uint16
+--rw bgp-attribute-list
|   +--rw bgp-origin?      enumeration
|   +--rw bgp-aspath
|   |   +--rw usascount?    uint16
|   |   +--rw ulrefcount?  uint16
|   |   +--rw asstring?    string
|   |   +--ro usascountfrcomp? uint16
|   |   +--rw usastotalcount? uint16
|   |   +--rw usas4pathlen? uint16
|   |   +--rw as4pathvalue? string
+--rw bgp-nexthop?      inet:ip-address

```

```

+---rw bgp-med?                                uint32
+---rw bgp-localpref?                          uint32
+---rw bgp-atomic-aggregate?                  uint32
+---rw bgp-aggregator
|   +---rw ulipaddress?                        uint32
|   +---rw ulasnumber?                        uint32
+---rw bgp-commattr
|   +---rw uscommsize?                        uint16
|   +---rw ulrefcomm?                        uint32
|   +---rw commattr-value?                    string
+---rw bgp-extcommattr
|   +---rw custom-community
|   |   +---rw valid                        boolean
|   |   +---rw insertion-point             uint32
|   |   +---rw community-id                uint8
|   |   +---rw cost-id                     uint32
|   +---rw usextcommsize?                    uint16
|   +---rw ulrefcount?                      uint32
|   +---rw extcommattr-value?                string
+---rw bgp-clusterlist
|   +---rw uscluslen?                        uint16
|   +---rw ulrefclus?                       uint32
|   +---rw clusterlist-value?                string
+---rw bgp-originator-id?                    uint32
+---ro bgp-route-create?                      enumeration
+---ro bgp-rt-state-info
|   +---ro rib-current-state?                rib-state-def
|   +---ro rib-last-state?                  rib-state-def
|   +---ro rib-rejected-reason?              enumeration
|   +---ro not-preferred-reason?             enumeration
+---rw bgp-ipv6-uni-instance
|   +---rw bgp-instance-name?                string
|   +---rw bgp-instance-create?              enumeration
|   +---rw bgp-instance-type?                enumeration
|   +---rw bgp-vendor-type?                  enumeration
|   +---rw afi?                              afi-def
|   +---rw safi?                              safi-def
+---rw bgp-local-rib
|   +---rw bgp-route-list* [ipv6-route ipv6-prefix-length]
|   |   +---rw ipv6-route                    inet:ipv6-prefix
|   |   +---rw ipv6-prefix-length            uint8
|   |   +---rw bgp-route-type?                enumeration
|   |   +---rw route-admin-distance          uint16
|   |   +---rw bgp-attribute-list
|   |   |   +---rw bgp-origin?                enumeration
|   |   |   +---rw bgp-aspath
|   |   |   |   +---rw usascount?              uint16
|   |   |   |   +---rw ulrefcount?              uint16

```

```

| | | +--rw asstring? string
| | | +---ro usascountfrcomp? uint16
| | | +---rw usastotalcount? uint16
| | | +---rw usas4pathlen? uint16
| | | +---rw as4pathvalue? string
+---rw bgp-nexthop? inet:ip-address
+---rw bgp-med? uint32
+---rw bgp-localpref? uint32
+---rw bgp-atomic-aggregate? uint32
+---rw bgp-aggregator
| | +---rw ulipaddress? uint32
| | +---rw ulasnumber? uint32
+---rw bgp-commattr
| | +---rw uscommsize? uint16
| | +---rw ulrefcomm? uint32
| | +---rw commattr-value? string
+---rw bgp-extcommattr
| | +---rw custom-community
| | | +---rw valid boolean
| | | +---rw insertion-point uint32
| | | +---rw community-id uint8
| | | +---rw cost-id uint32
+---rw usextcommsize? uint16
+---rw ulrefcount? uint32
+---rw extcommattr-value? string
+---rw bgp-clusterlist
| | +---rw uscluslen? uint16
| | +---rw ulrefclust? uint32
| | +---rw clusterlist-value? string
+---rw bgp-originator-id? uint32
+---ro bgp-route-create? enumeration
+---ro bgp-rt-state-info
+---ro rib-current-state? rib-state-def
+---ro rib-last-state? rib-state-def
+---ro rib-rejected-reason? enumeration
+---ro not-preferred-reason? enumeration
+---rw bgp-peer-list* [bgp-peer-name]
+---rw peer-session-address
| | +---rw local-ipv6-address? inet:ipv6-prefix
| | +---rw remote-ipv6-address? inet:ipv6-prefix
+---rw bgp-peer-name string
+---ro bgp-peer-type? enumeration
+---ro bgp-peer-create? enumeration
+---rw bgp-policy-in
+---rw bgp-policy-out
+---ro peer-policy-in-pset
+---ro peer-policy-out-pset
+---rw peer-state-info

```



```

+--ro peer-current-state?          peer-state
+--ro peer-last-state?             peer-state
+--ro peer-down-reason
|   +--ro error-code?              enumeration
|   +--ro sub-error-code
|       +--ro (sub-error-code-type)?
|           +---:(head-error-sub-code)
|               |   +--ro head-error-sub-value?  enumeration
|           +---:(open-error-sub-code)
|               |   +--ro open-error-sub-value?  enumeration
|           +---:(update-error-sub-code)
|               |   +--ro update-error-sub-value? enumeration
|           +---:(route-refresh-error-sub-code)
|               +--ro route-refresh-error-sub-value?
|                   enumeration
+--ro peer-received-update-cnt?    uint64
+--ro peer-transmit-update-cnt?    uint64
+--ro peer-received-route-cnt?     uint64
+--ro peer-send-route-cnt?         uint64
+--rw max-prefix-rcv-limit?        uint64
+--rw max-prefix-xmt-limit?        uint64
+--ro peer-prefix-high?            uint64
+--ro peer-prefix-low?             uint64
+--ro peer-perfix-ave?             uint64
+--ro peer-prefix-time?            uint64
+--ro peer-prefix-max?             uint64
+--ro publish-version?             uint64
+--rw bgp-rib-in
+--rw bgp-rib-in-list* [ipv6-route ipv6-prefix-length]
|   +--rw ipv6-route                inet:ipv6-prefix
|   +--rw ipv6-prefix-length        uint8
|   +--rw bgp-route-type?           enumeration
|   +--rw route-admin-distance      uint16
|   +--rw bgp-attribute-list
|       |   +--rw bgp-origin?        enumeration
|       |   +--rw bgp-aspath
|       |       |   +--rw usascount?    uint16
|       |       |   +--rw ulrefcount?   uint16
|       |       |   +--rw asstring?     string
|       |       |   +--ro usascountfrcomp? uint16
|       |       |   +--rw usastotalcount? uint16
|       |       |   +--rw usas4pathlen? uint16
|       |       |   +--rw as4pathvalue? string
|       |   +--rw bgp-nexthop?    inet:ip-address
|       |   +--rw bgp-med?         uint32
|       |   +--rw bgp-localpref?   uint32
|       |   +--rw bgp-atomic-aggregate? uint32
|       |   +--rw bgp-aggregator

```

```

|--rw ulipaddress?          uint32
|--rw ulasnumber?          uint32
+--rw bgp-commattr
|   |--rw uscommssize?      uint16
|   |--rw ulrefcomm?       uint32
|   |--rw commattr-value?   string
+--rw bgp-extcommattr
|   |--rw custom-community
|   |   |--rw valid         boolean
|   |   |--rw insertion-point uint32
|   |   |--rw community-id  uint8
|   |   |--rw cost-id       uint32
|   |--rw usextcommssize?   uint16
|   |--rw ulrefcount?      uint32
|   |--rw extcommattr-value? string
+--rw bgp-clusterlist
|   |--rw uscluslen?        uint16
|   |--rw ulrefclus?       uint32
|   |--rw clusterlist-value? string
+--rw bgp-originator-id?   uint32
+--ro bgp-route-create?     enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state?  rib-state-def
|   +--ro rib-last-state?     rib-state-def
|   +--ro rib-rejected-reason? enumeration
|   +--ro not-preferred-reason? enumeration
+--rw bgp-rib-out
|   +--rw bgp-rib-out-list* [ipv6-route ipv6-prefix-length]
|   |--rw ipv6-route          inet:ipv6-prefix
|   |--rw ipv6-prefix-length  uint8
|   |--rw bgp-route-type?     enumeration
|   |--rw route-admin-distance uint16
|   +--rw bgp-attribute-list
|   |   +--rw bgp-origin?      enumeration
|   |   +--rw bgp-aspath
|   |   |   |--rw usascount?    uint16
|   |   |   |--rw ulrefcount?   uint16
|   |   |   |--rw asstring?     string
|   |   |   |--ro usascountfrcomp? uint16
|   |   |   |--rw usastotalcount? uint16
|   |   |   |--rw usas4pathlen?  uint16
|   |   |   |--rw as4pathvalue?  string
|   |--rw bgp-nexthop?        et:ip-address
|   |--rw bgp-med?            uint32
|   |--rw bgp-localpref?      uint32
|   |--rw bgp-atomic-aggregate? uint32
|   +--rw bgp-aggregator
|   |   |--rw ulipaddress?      uint32

```

```

| | | | | +--rw ulasnumber?          uint32
| | | | | +---rw bgp-commattr
| | | | | | +--rw uscommsize?        uint16
| | | | | | +--rw ulrefcomm?         uint32
| | | | | | +--rw commattr-value?    string
| | | | | +---rw bgp-extcommattr
| | | | | | +--rw custom-community
| | | | | | | +--rw valid              boolean
| | | | | | | +--rw insertion-point   uint32
| | | | | | | +--rw community-id      uint8
| | | | | | | +--rw cost-id           uint32
| | | | | | +--rw usextcommsize?     uint16
| | | | | | +--rw ulrefcount?        uint32
| | | | | | +--rw extcommattr-value?  string
| | | | | +---rw bgp-clusterlist
| | | | | | +--rw uscluslen?          uint16
| | | | | | +--rw ulrefclus?         uint32
| | | | | | +--rw clusterlist-value?  string
| | | | | +---rw bgp-originator-id?   uint32
| | | | +---ro bgp-route-create?       enumeration
| | | +---ro bgp-rt-state-info
| | | | +--ro rib-current-state?       rib-state-def
| | | | +--ro rib-last-state?         rib-state-def
| | | | +--ro rib-rejected-reason?     enumeration
| | | | +--ro not-preferred-reason?    enumeration
+---rw bgp-labeled-ipv6-instance
| +---rw bgp-instance-name?           string
| +---rw bgp-instance-create?         enumeration
| +---rw bgp-instance-type?           enumeration
| +---rw bgp-vendor-type?             enumeration
| +---rw afi?                         afi-def
| +---rw safi?                        safi-def
+---rw bgp-local-rib
| | +---rw bgp-route-list*
| | | [ipv6-route ipv6-prefix-length mpls-label]
| | | +---rw ipv6-route                inet:ipv6-prefix
| | | +---rw ipv6-prefix-length         uint8
| | | +---rw mpls-label                 mpls-label-def
| | | +---rw bgp-route-type?           enumeration
| | | +---rw route-admin-distance      uint16
| | | +---rw bgp-attribute-list
| | | | +---rw bgp-origin?             enumeration
| | | | +---rw bgp-aspath
| | | | | +---rw usascount?            uint16
| | | | | +---rw ulrefcount?          uint16
| | | | | +---rw asstring?            string
| | | | | +--ro usascountfrcomp?      uint16
| | | | | +---rw usastotalcount?      uint16

```

```

| | | | | +--rw usas4pathlen?      uint16
| | | | | +--rw as4pathvalue?    string
| | | | | +--rw bgp-nexthop?  inet:ip-address
| | | | | +--rw bgp-med?          uint32
| | | | | +--rw bgp-localpref?    uint32
| | | | | +--rw bgp-atomic-aggregate? uint32
| | | | | +--rw bgp-aggregator
| | | | | | +--rw ulipaddress?    uint32
| | | | | | +--rw ulasnumber?    uint32
| | | | | +--rw bgp-commattr
| | | | | | +--rw uscommsize?      uint16
| | | | | | +--rw ulrefcomm?      uint32
| | | | | | +--rw commattr-value? string
| | | | | +--rw bgp-extcommattr
| | | | | | +--rw custom-community
| | | | | | | +--rw valid          boolean
| | | | | | | +--rw insertion-point uint32
| | | | | | | +--rw community-id  uint8
| | | | | | | +--rw cost-id       uint32
| | | | | | +--rw usextcommsize?  uint16
| | | | | | +--rw ulrefcount?     uint32
| | | | | | +--rw extcommattr-value? string
| | | | | +--rw bgp-clusterlist
| | | | | | +--rw uscluslen?      uint16
| | | | | | +--rw ulrefclus?     uint32
| | | | | | +--rw clusterlist-value? string
| | | | | +--rw bgp-originator-id? uint32
| | | | +--ro bgp-route-create?    enumeration
| | | | +--ro bgp-rt-state-info
| | | | | +--ro rib-current-state? rib-state-def
| | | | | +--ro rib-last-state?   rib-state-def
| | | | | +--ro rib-rejected-reason? enumeration
| | | | | +--ro not-preferred-reason? enumeration
| | | +--rw bgp-peer-list* [bgp-peer-name]
| | | | +--rw peer-session-address
| | | | | +--rw local-ipv6-address? inet:ipv6-prefix
| | | | | +--rw remote-ipv6-address? inet:ipv6-prefix
| | | | +--rw bgp-peer-name        string
| | | | +--ro bgp-peer-type?        enumeration
| | | | +--ro bgp-peer-create?      enumeration
| | | | +--rw bgp-policy-in
| | | | +--rw bgp-policy-out
| | | | +--ro peer-policy-in-pset
| | | | +--ro peer-policy-out-pset
| | | | +--rw peer-state-info
| | | | | +--ro peer-current-state? peer-state
| | | | | +--ro peer-last-state?   peer-state
| | | | +--ro peer-down-reason

```

```

+--ro error-code?          enumeration
+--ro sub-error-code
  +--ro (sub-error-code-type)?
    +--:(head-error-sub-code)
      | +--ro head-error-sub-value? enumeration
    +--:(open-error-sub-code)
      | +--ro open-error-sub-value? enumeration
    +--:(update-error-sub-code)
      | +--ro update-error-sub-value? enumeration
    +--:(route-refresh-error-sub-code)
      +--ro route-refresh-error-sub-value?
        enumeration
+--ro peer-received-update-cnt?  uint64
+--ro peer-transmit-update-cnt?  uint64
+--ro peer-received-route-cnt?   uint64
+--ro peer-send-route-cnt?       uint64
+--rw max-prefix-rcv-limit?      uint64
+--rw max-prefix-xmt-limit?      uint64
+--ro peer-prefix-high?          uint64
+--ro peer-prefix-low?           uint64
+--ro peer-perfix-ave?            uint64
+--ro peer-prefix-time?          uint64
+--ro peer-prefix-max?           uint64
+--ro publish-version?           uint64
+--rw bgp-rib-in
  +--rw bgp-rib-in-list*
    [ipv6-route ipv6-prefix-length mpls-label]
    +--rw ipv6-route              inet:ipv6-prefix
    +--rw ipv6-prefix-length      uint8
    +--rw mpls-label              mpls-label-def
    +--rw bgp-route-type?         enumeration
    +--rw route-admin-distance    uint16
    +--rw bgp-attribute-list
      +--rw bgp-origin?           enumeration
      +--rw bgp-aspath
        +--rw usascount?          uint16
        +--rw ulrefcount?         uint16
        +--rw asstring?           string
        +--ro usascountfrcomp?    uint16
        +--rw usastotalcount?     uint16
        +--rw usas4pathlen?       uint16
        +--rw as4pathvalue?       string
      +--rw bgp-nexthop?          inet:ip-address
      +--rw bgp-med?              uint32
      +--rw bgp-localpref?        uint32
      +--rw bgp-atomic-aggregate? uint32
      +--rw bgp-aggregator
        +--rw ulipaddress?        uint32

```

```

|--rw ulasumber?      uint32
+--rw bgp-commattr
|   +--rw uscommssize?      uint16
|   +--rw ulrefcomm?      uint32
|   +--rw commattr-value?  string
+--rw bgp-extcommattr
|   +--rw custom-community
|   |   +--rw valid        boolean
|   |   +--rw insertion-point uint32
|   |   +--rw community-id  uint8
|   |   +--rw cost-id       uint32
|   +--rw usextcommssize?   uint16
|   +--rw ulrefcount?       uint32
|   +--rw extcommattr-value? string
+--rw bgp-clusterlist
|   +--rw uscluslen?        uint16
|   +--rw ulrefclus?        uint32
|   +--rw clusterlist-value? string
+--rw bgp-originator-id?    uint32
+--ro bgp-route-create?      enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state?  rib-state-def
|   +--ro rib-last-state?     rib-state-def
|   +--ro rib-rejected-reason? enumeration
|   +--ro not-preferred-reason? enumeration
+--rw bgp-rib-out
|   +--rw bgp-rib-out-list*
|   |   [ipv6-route ipv6-prefix-length mpls-label]
|   |   +--rw ipv6-route      inet:ipv6-prefix
|   |   +--rw ipv6-prefix-length uint8
|   |   +--rw mpls-label      mpls-label-def
|   |   +--rw bgp-route-type?  enumeration
|   |   +--rw route-admin-distance uint16
|   |   +--rw bgp-attribute-list
|   |   |   +--rw bgp-origin?      enumeration
|   |   |   +--rw bgp-aspath
|   |   |   |   +--rw usascount?      uint16
|   |   |   |   +--rw ulrefcount?     uint16
|   |   |   |   +--rw asstring?      string
|   |   |   |   +--ro usascountfrcomp? uint16
|   |   |   |   +--rw usastotalcount? uint16
|   |   |   |   +--rw usas4pathlen?   uint16
|   |   |   |   +--rw as4pathvalue?   string
|   |   +--rw bgp-nexthop?      inet:ip-address
|   |   +--rw bgp-med?          uint32
|   |   +--rw bgp-localpref?    uint32
|   |   +--rw bgp-atomic-aggregate? uint32
|   |   +--rw bgp-aggregator

```

```

|--rw ulipaddress?      uint32
|--rw ulasnumber?      uint32
+--rw bgp-commattr
|   |--rw uscommsize?      uint16
|   |--rw ulrefcomm?      uint32
|   |--rw commattr-value?  string
+--rw bgp-extcommattr
|   |--rw custom-community
|   |   |--rw valid          boolean
|   |   |--rw insertion-point uint32
|   |   |--rw community-id   uint8
|   |   |--rw cost-id        uint32
|   |--rw usextcommsize?    uint16
|   |--rw ulrefcount?      uint32
|   |--rw extcommattr-value? string
+--rw bgp-clusterlist
|   |--rw uscluslen?      uint16
|   |--rw ulrefclus?      uint32
|   |--rw clusterlist-value? string
+--rw bgp-originator-id?  uint32
+--ro bgp-route-create?    enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state?  rib-state-def
|   +--ro rib-last-state?     rib-state-def
|   +--ro rib-rejected-reason? enumeration
|   +--ro not-preferred-reason? enumeration
+--rw bgp-ipv6-multi-instance
|   +--rw bgp-instance-name?  string
|   +--rw bgp-instance-create? enumeration
|   +--rw bgp-instance-type?  enumeration
|   +--rw bgp-vendor-type?    enumeration
|   +--rw afi?                afi-def
|   +--rw safi?                safi-def
+--rw bgp-local-rib
|   +--rw bgp-route-list* [ipv6-route ipv6-prefix-length]
|   |   +--rw ipv6-route      inet:ipv6-prefix
|   |   +--rw ipv6-prefix-length uint8
|   |   +--rw bgp-route-type?  enumeration
|   |   +--rw route-admin-distance uint16
|   |   +--rw bgp-attribute-list
|   |   |   +--rw bgp-origin?      enumeration
|   |   |   +--rw bgp-aspath
|   |   |   |   +--rw usascount?    uint16
|   |   |   |   +--rw ulrefcount?    uint16
|   |   |   |   +--rw asstring?      string
|   |   |   |   +--ro usascountfrcomp? uint16
|   |   |   |   +--rw usastotalcount? uint16
|   |   |   |   +--rw usas4pathlen?  uint16

```

```
| | +--rw as4pathvalue?      string
| | +---rw bgp-nexthop?      inet:ip-address
| | +---rw bgp-med?          uint32
| | +---rw bgp-localpref?    uint32
| | +---rw bgp-atomic-aggregate?  uint32
| | +---rw bgp-aggregator
| | |   +---rw ulipaddress?    uint32
| | |   +---rw ulasnumber?     uint32
| | +---rw bgp-commattr
| | |   +---rw uscommsize?      uint16
| | |   +---rw ulrefcomm?       uint32
| | |   +---rw commattr-value?  string
| | +---rw bgp-extcommattr
| | |   +---rw custom-community
| | |   |   +---rw valid        boolean
| | |   |   +---rw insertion-point  uint32
| | |   |   +---rw community-id   uint8
| | |   |   +---rw cost-id        uint32
| | |   +---rw usextcommsize?    uint16
| | |   +---rw ulrefcount?       uint32
| | |   +---rw extcommattr-value? string
| | +---rw bgp-clusterlist
| | |   +---rw uscluslen?        uint16
| | |   +---rw ulrefclust?       uint32
| | |   +---rw clusterlist-value? string
| | +---rw bgp-originator-id?    uint32
+---ro bgp-route-create?          enumeration
+---ro bgp-rt-state-info
|   +---ro rib-current-state?      rib-state-def
|   +---ro rib-last-state?         rib-state-def
|   +---ro rib-rejected-reason?    enumeration
|   +---ro not-preferred-reason?   enumeration
+---rw bgp-peer-list* [bgp-peer-name]
|   +---rw peer-session-address
|   |   +---rw local-ipv6-address?  inet:ipv6-prefix
|   |   +---rw remote-ipv6-address? inet:ipv6-prefix
+---rw bgp-peer-name              string
+---ro bgp-peer-type?             enumeration
+---ro bgp-peer-create?           enumeration
+---rw bgp-policy-in
+---rw bgp-policy-out
+---ro peer-policy-in-pset
+---ro peer-policy-out-pset
+---rw peer-state-info
|   +---ro peer-current-state?      peer-state
|   +---ro peer-last-state?         peer-state
|   +---ro peer-down-reason
|   |   +---ro error-code?          enumeration
```



```

| | | +--ro sub-error-code
| | |   +--ro (sub-error-code-type)?
| | |     +--:(head-error-sub-code)
| | |       | +--ro head-error-sub-value? enumeration
| | |     +--:(open-error-sub-code)
| | |       | +--ro open-error-sub-value? enumeration
| | |     +--:(update-error-sub-code)
| | |       | +--ro update-error-sub-value?
| | |         enumeration
| | |   +--:(route-refresh-error-sub-code)
| | |     +--ro route-refresh-error-sub-value?
| | |       enumeration
|
| +--ro peer-received-update-cnt? uint64
| +--ro peer-transmit-update-cnt? uint64
| +--ro peer-received-route-cnt?  uint64
| +--ro peer-send-route-cnt?     uint64
| +--rw max-prefix-rcv-limit?    uint64
| +--rw max-prefix-xmt-limit?    uint64
| +--ro peer-prefix-high?        uint64
| +--ro peer-prefix-low?         uint64
| +--ro peer-perfix-ave?          uint64
| +--ro peer-prefix-time?        uint64
| +--ro peer-prefix-max?         uint64
| +--ro publish-version?         uint64
+--rw bgp-rib-in
+--rw bgp-rib-in-list* [ipv6-route ipv6-prefix-length]
+--rw ipv6-route            inet:ipv6-prefix
+--rw ipv6-prefix-length    uint8
+--rw bgp-route-type?       enumeration
+--rw route-admin-distance  uint16
+--rw bgp-attribute-list
| +--rw bgp-origin?          enumeration
| +--rw bgp-aspath
| | +--rw usascount?          uint16
| | +--rw ulrefcount?         uint16
| | +--rw asstring?           string
| | +--ro usascountfrcomp?    uint16
| | +--rw usastotalcount?     uint16
| | +--rw usas4pathlen?       uint16
| | +--rw as4pathvalue?       string
| +--rw bgp-nexthop?         inet:ip-address
| +--rw bgp-med?              uint32
| +--rw bgp-localpref?       uint32
| +--rw bgp-atomic-aggregate? uint32
| +--rw bgp-aggregator
| | +--rw ulipaddress?        uint32
| | +--rw ulasnumber?         uint32
+--rw bgp-commattr

```

```

|--rw uscommsize?          uint16
+--rw ulrefcomm?          uint32
+--rw commattr-value?     string
+--rw bgp-extcommattr
| +--rw custom-community
| | +--rw valid            boolean
| | +--rw insertion-point uint32
| | +--rw community-id    uint8
| | +--rw cost-id         uint32
| +--rw usextcommsize?    uint16
| +--rw ulrefcount?       uint32
| +--rw extcommattr-value? string
+--rw bgp-clusterlist
| +--rw uscluslen?        uint16
| +--rw ulrefclus?        uint32
| +--rw clusterlist-value? string
+--rw bgp-originator-id?  uint32
+--ro bgp-route-create?    enumeration
+--ro bgp-rt-state-info
| +--ro rib-current-state?  rib-state-def
| +--ro rib-last-state?    rib-state-def
| +--ro rib-rejected-reason? enumeration
| +--ro not-preferred-reason? enumeration
+--rw bgp-rib-out
+--rw bgp-rib-out-list* [ipv6-route ipv6-prefix-length]
| +--rw ipv6-route          inet:ipv6-prefix
| +--rw ipv6-prefix-length  uint8
| +--rw bgp-route-type?     enumeration
| +--rw route-admin-distance uint16
| +--rw bgp-attribute-list
| | +--rw bgp-origin?       enumeration
| | +--rw bgp-aspath
| | | +--rw usascount?      uint16
| | | +--rw ulrefcount?     uint16
| | | +--rw asstring?       string
| | | +--ro usascountfrcomp? uint16
| | | +--rw usastotalcount? uint16
| | | +--rw usas4pathlen?   uint16
| | | +--rw as4pathvalue?   string
| +--rw bgp-nexthop?        inet:ip-address
| +--rw bgp-med?            uint32
| +--rw bgp-localpref?      uint32
| +--rw bgp-atomic-aggregate? uint32
| +--rw bgp-aggregator
| | +--rw ulipaddress?      uint32
| | +--rw ulasnumber?       uint32
+--rw bgp-commattr
| +--rw uscommsize?        uint16

```

```

| | | | | +--rw ulrefcomm?          uint32
| | | | | +--rw commattr-value?   string
+--rw bgp-extcommattr
| | | | | +--rw custom-community
| | | | | | | +--rw valid              boolean
| | | | | | | +--rw insertion-point  uint32
| | | | | | | +--rw community-id     uint8
| | | | | | | +--rw cost-id         uint32
+--rw usextcommsize?      uint16
+--rw ulrefcount?        uint32
+--rw extcommattr-value? string
+--rw bgp-clusterlist
| | | | | +--rw uscluslen?           uint16
| | | | | +--rw ulrefclus?         uint32
| | | | | +--rw clusterlist-value? string
+--rw bgp-originator-id? uint32
+--ro bgp-route-create?   enumeration
+--ro bgp-rt-state-info
| | | | | +--ro rib-current-state?    rib-state-def
| | | | | +--ro rib-last-state?      rib-state-def
+--ro rib-rejected-reason? enumeration
+--ro not-preferred-reason? enumeration
+--rw bgp-vpnv6-instance
| | | | | +--rw bgp-instance-name?   string
| | | | | +--rw bgp-instance-create? enumeration
| | | | | +--rw bgp-instance-type?   enumeration
| | | | | +--rw bgp-vendor-type?     enumeration
| | | | | +--rw afi?                 afi-def
| | | | | +--rw safi?                safi-def
+--rw bgp-local-rib
| | | | | +--rw bgp-route-list*
| | | | | [ipv6-route ipv6-prefix-length route-distinguisher]
| | | | | | | +--rw ipv6-route       inet:ipv6-prefix
| | | | | | | +--rw ipv6-prefix-length  uint8
| | | | | | | +--rw mpls-label?       mpls-label-def
+--rw route-distinguisher route-distinguisher-def
+--rw bgp-route-type?      enumeration
+--rw route-admin-distance uint16
+--rw bgp-attribute-list
| | | | | +--rw bgp-origin?          enumeration
| | | | | +--rw bgp-aspath
| | | | | | | +--rw usascount?        uint16
| | | | | | | +--rw ulrefcount?       uint16
| | | | | | | +--rw asstring?         string
+--ro usascountfrcomp?     uint16
+--ro usastotalcount?      uint16
+--rw usas4pathlen?        uint16
+--rw as4pathvalue?        string

```

```

+---rw bgp-nexthop?                inet:ip-address
+---rw bgp-med?                     uint32
+---rw bgp-localpref?               uint32
+---rw bgp-atomic-aggregate?       uint32
+---rw bgp-aggregator
|   +---rw ulipaddress?             uint32
|   +---rw ulasnumber?              uint32
+---rw bgp-commattr
|   +---rw uscommsize?               uint16
|   +---rw ulrefcomm?                uint32
|   +---rw commattr-value?          string
+---rw bgp-extcommattr
|   +---rw custom-community
|   |   +---rw valid                 boolean
|   |   +---rw insertion-point      uint32
|   |   +---rw community-id         uint8
|   |   +---rw cost-id              uint32
|   +---rw usextcommsize?           uint16
|   +---rw ulrefcount?              uint32
|   +---rw extcommattr-value?       string
+---rw bgp-clusterlist
|   +---rw uscluslen?                uint16
|   +---rw ulrefclus?               uint32
|   +---rw clusterlist-value?       string
+---rw bgp-originator-id?           uint32
+---ro bgp-route-create?             enumeration
+---ro bgp-rt-state-info
|   +---ro rib-current-state?        rib-state-def
|   +---ro rib-last-state?           rib-state-def
|   +---ro rib-rejected-reason?      enumeration
|   +---ro not-preferred-reason?     enumeration
+---rw bgp-peer-list* [bgp-peer-name]
+---rw peer-session-address
|   +---rw local-ipv6-address?       inet:ipv6-prefix
|   +---rw remote-ipv6-address?     inet:ipv6-prefix
+---rw bgp-peer-name                 string
+---ro bgp-peer-type?                enumeration
+---ro bgp-peer-create?              enumeration
+---rw bgp-policy-in
+---rw bgp-policy-out
+---ro peer-policy-in-pset
+---ro peer-policy-out-pset
+---rw peer-state-info
|   +---ro peer-current-state?        peer-state
|   +---ro peer-last-state?           peer-state
|   +---ro peer-down-reason
|   |   +---ro error-code?            enumeration
|   |   +---ro sub-error-code

```

```

| | +--ro (sub-error-code-type)?
| | +--:(head-error-sub-code)
| | | +--ro head-error-sub-value? enumeration
| | +--:(open-error-sub-code)
| | | +--ro open-error-sub-value? enumeration
| | +--:(update-error-sub-code)
| | | +--ro update-error-sub-value?
| | | enumeration
| | +--:(route-refresh-error-sub-code)
| | | +--ro route-refresh-error-sub-value?
| | | enumeration
+--ro peer-received-update-cnt? uint64
+--ro peer-transmit-update-cnt? uint64
+--ro peer-received-route-cnt? uint64
+--ro peer-send-route-cnt? uint64
+--rw max-prefix-rcv-limit? uint64
+--rw max-prefix-xmt-limit? uint64
+--ro peer-prefix-high? uint64
+--ro peer-prefix-low? uint64
+--ro peer-perfix-ave? uint64
+--ro peer-prefix-time? uint64
+--ro peer-prefix-max? uint64
+--ro publish-version? uint64
+--rw bgp-rib-in
| +--rw bgp-rib-in-list*
| [ipv6-route ipv6-prefix-length route-distinguisher]
| | +--rw ipv6-route inet:ipv6-prefix
| | +--rw ipv6-prefix-length uint8
| | +--rw mpls-label? mpls-label-def
| | +--rw route-distinguisher route-distinguisher-def
| | +--rw bgp-route-type? enumeration
| | +--rw route-admin-distance uint16
| | +--rw bgp-attribute-list
| | | +--rw bgp-origin? enumeration
| | | +--rw bgp-aspath
| | | | +--rw usascount? uint16
| | | | +--rw ulrefcount? uint16
| | | | +--rw asstring? string
| | | | +--ro usascountfrcomp? uint16
| | | | +--rw usastotalcount? uint16
| | | | +--rw usas4pathlen? uint16
| | | | +--rw as4pathvalue? string
| | | +--rw bgp-nexthop? inet:ip-address
| | | +--rw bgp-med? uint32
| | | +--rw bgp-localpref? uint32
| | | +--rw bgp-atOMIC-aggregate? uint32
| | | +--rw bgp-aggregator
| | | | +--rw ulipaddress? uint32

```

```

|--rw ulasnumber?          uint32
+--rw bgp-commattr
|   |--rw uscommsize?      uint16
|   |--rw ulrefcomm?      uint32
|   |--rw commattr-value?  string
+--rw bgp-extcommattr
|   |--rw custom-community
|   |   |--rw valid        boolean
|   |   |--rw insertion-point uint32
|   |   |--rw community-id  uint8
|   |   |--rw cost-id       uint32
|   |--rw usextcommsize?   uint16
|   |--rw ulrefcount?      uint32
|   |--rw extcommattr-value? string
+--rw bgp-clusterlist
|   |--rw uscluslen?       uint16
|   |--rw ulrefclus?       uint32
|   |--rw clusterlist-value? string
+--rw bgp-originator-id?   uint32
+--ro bgp-route-create?     enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state?    rib-state-def
|   +--ro rib-last-state?       rib-state-def
|   +--ro rib-rejected-reason?  enumeration
|   +--ro not-preferred-reason? enumeration
+--rw bgp-rib-out
+--rw bgp-rib-out-list*
[ipv6-route ipv6-prefix-length route-distinguisher]
+--rw ipv6-route          inet:ipv6-prefix
+--rw ipv6-prefix-length  uint8
+--rw mpls-label?         mpls-label-def
+--rw route-distinguisher route-distinguisher-def
+--rw bgp-route-type?     enumeration
+--rw route-admin-distance uint16
+--rw bgp-attribute-list
|   +--rw bgp-origin?      enumeration
|   +--rw bgp-aspath
|   |   |--rw usascount?    uint16
|   |   |--rw ulrefcount?   uint16
|   |   |--rw asstring?     string
|   |   |--ro usascountfrcomp? uint16
|   |   |--rw usastotalcount? uint16
|   |   |--rw usas4pathlen?  uint16
|   |   |--rw as4pathvalue?  string
|   +--rw bgp-nexthop?     inet:ip-address
|   +--rw bgp-med?         uint32
|   +--rw bgp-localpref?   uint32
|   +--rw bgp-atomic-aggregate? uint32

```

```

+---rw bgp-aggregator
|   +---rw ulipaddress?      uint32
|   +---rw ulasnumber?      uint32
+---rw bgp-commattr
|   +---rw uscommsize?       uint16
|   +---rw ulrefcomm?       uint32
|   +---rw commattr-value?   string
+---rw bgp-extcommattr
|   +---rw custom-community
|   |   +---rw valid         boolean
|   |   +---rw insertion-point uint32
|   |   +---rw community-id   uint8
|   |   +---rw cost-id        uint32
|   +---rw usextcommsize?    uint16
|   +---rw ulrefcount?       uint32
|   +---rw extcommattr-value? string
+---rw bgp-clusterlist
|   +---rw uscluslen?        uint16
|   +---rw ulrefclus?        uint32
|   +---rw clusterlist-value? string
+---rw bgp-originator-id?    uint32
+---ro bgp-route-create?      enumeration
+---ro bgp-rt-state-info
|   +---ro rib-current-state?  rib-state-def
|   +---ro rib-last-state?     rib-state-def
|   +---ro rib-rejected-reason? enumeration
|   +---ro not-preferred-reason? enumeration
+---rw bgp-ipv6-l3vpn-instance-list* [bgp-l3vpn-instance-name]
|   +---rw bgp-l3vpn-instance-name string
|   +---rw bgp-instance-name?      string
|   +---rw bgp-instance-create?    enumeration
|   +---rw bgp-instance-type?      enumeration
|   +---rw bgp-vendor-type?        enumeration
|   +---rw afi?                    afi-def
|   +---rw safi?                    safi-def
+---rw bgp-local-rib
|   +---rw bgp-route-list* [ipv6-route ipv6-prefix-length]
|   |   +---rw ipv6-route          inet:ipv6-prefix
|   |   +---rw ipv6-prefix-length  uint8
|   |   +---rw bgp-route-type?     enumeration
|   |   +---rw route-admin-distance uint16
|   +---rw bgp-attribute-list
|   |   +---rw bgp-origin?          enumeration
|   |   +---rw bgp-aspath
|   |   |   +---rw usascount?       uint16
|   |   |   +---rw ulrefcount?      uint16
|   |   |   +---rw asstring?        string
|   |   |   +---ro usascountfrcomp? uint16

```

```

| | +--rw usastotalcount?      uint16
| | +--rw usas4pathlen?       uint16
| | +--rw as4pathvalue?       string
+--rw bgp-nexthop?            inet:ip-address
+--rw bgp-med?                uint32
+--rw bgp-localpref?          uint32
+--rw bgp-atomic-aggregate?   uint32
+--rw bgp-aggregator
| | +--rw ulipaddress?        uint32
| | +--rw ulasnumber?         uint32
+--rw bgp-commattr
| | +--rw uscommsize?          uint16
| | +--rw ulrefcomm?          uint32
| | +--rw commattr-value?     string
+--rw bgp-extcommattr
| | +--rw custom-community
| | +--rw valid                boolean
| | | +--rw insertion-point    uint32
| | | +--rw community-id       uint8
| | | +--rw cost-id            uint32
+--rw usextcommsize?          uint16
+--rw ulrefcount?             uint32
+--rw extcommattr-value?      string
+--rw bgp-clusterlist
| | +--rw uscluslen?           uint16
| | +--rw ulrefclus?          uint32
| | +--rw clusterlist-value?   string
+--rw bgp-originator-id?      uint32
+--ro bgp-route-create?        enumeration
+--ro bgp-rt-state-info
+--ro rib-current-state?       rib-state-def
+--ro rib-last-state?          rib-state-def
+--ro rib-rejected-reason?     enumeration
+--ro not-preferred-reason?    enumeration
+--rw bgp-peer-list* [bgp-peer-name]
+--rw peer-session-address
| | +--rw local-ipv6-address?   inet:ipv6-prefix
| | +--rw remote-ipv6-address?  inet:ipv6-prefix
+--rw bgp-peer-name            string
+--ro bgp-peer-type?           enumeration
+--ro bgp-peer-create?         enumeration
+--rw bgp-policy-in
+--rw bgp-policy-out
+--ro peer-policy-in-pset
+--ro peer-policy-out-pset
+--rw peer-state-info
| | +--ro peer-current-state?    peer-state
| | +--ro peer-last-state?      peer-state

```



```

+--ro peer-down-reason
+--ro error-code? enumeration
+--ro sub-error-code
+--ro (sub-error-code-type)?
+--:(head-error-sub-code)
| +--ro head-error-sub-value? enumeration
+--:(open-error-sub-code)
| +--ro open-error-sub-value? enumeration
+--:(update-error-sub-code)
| +--ro update-error-sub-value?
| enumeration
+--:(route-refresh-error-sub-code)
+--ro route-refresh-error-sub-value?
| enumeration
+--ro peer-received-update-cnt? uint64
+--ro peer-transmit-update-cnt? uint64
+--ro peer-received-route-cnt? uint64
+--ro peer-send-route-cnt? uint64
+--rw max-prefix-rcv-limit? uint64
+--rw max-prefix-xmt-limit? uint64
+--ro peer-prefix-high? uint64
+--ro peer-prefix-low? uint64
+--ro peer-perfix-ave? uint64
+--ro peer-prefix-time? uint64
+--ro peer-prefix-max? uint64
+--ro publish-version? uint64
+--rw bgp-rib-in
+--rw bgp-rib-in-list* [ipv6-route ipv6-prefix-length]
+--rw ipv6-route inet:ipv6-prefix
+--rw ipv6-prefix-length uint8
+--rw bgp-route-type? enumeration
+--rw route-admin-distance uint16
+--rw bgp-attribute-list
+--rw bgp-origin? enumeration
+--rw bgp-aspath
+--rw usascount? uint16
+--rw ulrefcount? uint16
+--rw asstring? string
+--ro usascountfrcomp? uint16
+--rw usastotalcount? uint16
+--rw usas4pathlen? uint16
+--rw as4pathvalue? string
+--rw bgp-nexthop? inet:ip-address
+--rw bgp-med? uint32
+--rw bgp-localpref? uint32
+--rw bgp-atomic-aggregate? uint32
+--rw bgp-aggregator
+--rw ulipaddress? uint32

```

```
+---rw ulasnumber?      uint32
+--rw bgp-commattr
|   +---rw uscommsize?      uint16
|   +---rw ulrefcomm?      uint32
|   +---rw commattr-value?  string
+--rw bgp-extcommattr
|   +---rw custom-community
|       |   +---rw valid          boolean
|       |   +---rw insertion-point uint32
|       |   +---rw community-id    uint8
|       |   +---rw cost-id         uint32
|   +---rw usextcommsize?      uint16
|   +---rw ulrefcount?         uint32
|   +---rw extcommattr-value?  string
+--rw bgp-clusterlist
|   +---rw uscluslen?          uint16
|   +---rw ulrefclust?        uint32
|   +---rw clusterlist-value?  string
+--rw bgp-originator-id?      uint32
+--ro bgp-route-create?        enumeration
+--ro bgp-rt-state-info
|   +---ro rib-current-state?    rib-state-def
|   +---ro rib-last-state?       rib-state-def
|   +---ro rib-rejected-reason?  enumeration
|   +---ro not-preferred-reason? enumeration
+--rw bgp-rib-out
|   +---rw bgp-rib-out-list*
|       [ipv6-route ipv6-prefix-length]
|   +---rw ipv6-route            inet:ipv6-prefix
|   +---rw ipv6-prefix-length    uint8
|   +---rw bgp-route-type?       enumeration
|   +---rw route-admin-distance  uint16
+--rw bgp-attribute-list
|   +---rw bgp-orign?            enumeration
|   +---rw bgp-aspath
|       |   +---rw usascount?      uint16
|       |   +---rw ulrefcount?     uint16
|       |   +---rw asstring?       string
|       |   +---ro usascountfrcomp? uint16
|       |   +---rw usastotalcount?  uint16
|       |   +---rw usas4pathlen?    uint16
|       |   +---rw as4pathvalue?    string
|   +---rw bgp-nexthop?          inet:ip-address
|   +---rw bgp-med?              uint32
|   +---rw bgp-localpref?        uint32
|   +---rw bgp-atOMIC-aggregate?  uint32
+--rw bgp-aggregator
|   +---rw ulipaddress?          uint32
```

```

| | | +--rw ulasnumber?      uint32
| | | +--rw bgp-commattr
| | | | +--rw uscommsize?      uint16
| | | | +--rw ulrefcomm?      uint32
| | | | +--rw commattr-value?  string
| | | +--rw bgp-extcommattr
| | | | +--rw custom-community
| | | | | +--rw valid          boolean
| | | | | +--rw insertion-point uint32
| | | | | +--rw community-id   uint8
| | | | | +--rw cost-id        uint32
| | | | +--rw usextcommsize?   uint16
| | | | +--rw ulrefcount?      uint32
| | | | +--rw extcommattr-value? string
| | | +--rw bgp-clusterlist
| | | | +--rw uscluslen?       uint16
| | | | +--rw ulrefclus?      uint32
| | | | +--rw clusterlist-value? string
| | | +--rw bgp-originator-id? uint32
| | +--ro bgp-route-create?    enumeration
| | +--ro bgp-rt-state-info
| | | +--ro rib-current-state?  rib-state-def
| | | +--ro rib-last-state?     rib-state-def
| | | +--ro rib-rejected-reason? enumeration
| | | +--ro not-preferred-reason? enumeration
+--rw bgp-ipv4-mvpn-instance
| | +--rw bgp-instance-name?    string
| | +--rw bgp-instance-create?  enumeration
| | +--rw bgp-instance-type?    enumeration
| | +--rw bgp-vendor-type?      enumeration
| | +--rw afi?                  afi-def
| | +--rw safi?                  safi-def
| | +--rw bgp-local-rib
| | | +--rw bgp-route-type?     enumeration
| | | +--rw route-admin-distance uint16
| | | +--rw bgp-attribute-list
| | | | +--rw bgp-origin?       enumeration
| | | | +--rw bgp-aspath
| | | | | +--rw usascount?       uint16
| | | | | +--rw ulrefcount?      uint16
| | | | | +--rw asstring?        string
| | | | | +--ro usascountfrcomp? uint16
| | | | | +--rw usastotalcount?  uint16
| | | | | +--rw usas4pathlen?    uint16
| | | | | +--rw as4pathvalue?    string
| | | +--rw bgp-nexthop?        inet:ip-address
| | | +--rw bgp-med?             uint32
| | | +--rw bgp-localpref?       uint32

```

```

+--rw bgp-atomic-aggregate?      uint32
+--rw bgp-aggregator
|   +--rw ulipaddress?      uint32
|   +--rw ulasnumber?      uint32
+--rw bgp-commattr
|   +--rw uscommsize?      uint16
|   +--rw ulrefcomm?      uint32
|   +--rw commattr-value?  string
+--rw bgp-extcommattr
|   +--rw custom-community
|   |   +--rw valid          boolean
|   |   +--rw insertion-point uint32
|   |   +--rw community-id   uint8
|   |   +--rw cost-id        uint32
|   +--rw usextcommsize?    uint16
|   +--rw ulrefcount?      uint32
|   +--rw extcommattr-value? string
+--rw bgp-clusterlist
|   +--rw uscluslen?      uint16
|   +--rw ulrefclus?      uint32
|   +--rw clusterlist-value? string
+--rw bgp-originator-id?      uint32
+--ro bgp-route-create?      enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state?    rib-state-def
|   +--ro rib-last-state?      rib-state-def
|   +--ro rib-rejected-reason?  enumeration
|   +--ro not-preferred-reason? enumeration
+--rw (bgp-mvpn-route-type)?
+--:(INTRA_AS_I_PMSI_AD)
|   +--rw intra-as-i-pmsi-ad-route*
[route-type route-distinguisher origin-router-ip-addr]
|   +--rw route-type          mvpn-route-type
|   +--rw route-distinguisher route-distinguisher-def
|   +--rw origin-router-ip-addr inet:ip-prefix
+--:(INTER_AS_I_PMSI_AD)
|   +--rw inter_as_i_pmsi_as_route*
[route-type route-distinguisher source-as]
|   +--rw route-type          mvpn-route-type
|   +--rw route-distinguisher route-distinguisher-def
|   +--rw source-as          inet:as-number
+--:(S_PMSI_AD)
|   +--rw s-pmsi-ad-route*
[route-type route-distinguisher multicast-source
multicast-group origin-router-ip-addr]
|   +--rw route-type          mvpn-route-type
|   +--rw route-distinguisher route-distinguisher-def
|   +--rw multicast-source    inet:ip-prefix

```

```

|         |--rw multicast-group          inet:ip-prefix
|         +--rw origin-router-ip-addr   inet:ip-prefix
|     +---:(LEAF_AD)
|         |--rw leaf-ad-route*
|             [route-type route-key origin-router-ip-addr]
|         +--rw route-type                mvpn-route-type
|         +--rw route-key                  binary
|         +--rw origin-router-ip-addr     inet:ip-prefix
|     +---:(SOURCE_ACTIVE_AD)
|         |--rw source-active-ad-route*
| [route-type route-distinguisher multicast-source multicast-group]
|         |--rw route-type                mvpn-route-type
|         |--rw route-distinguisher route-distinguisher-def
|         |--rw multicast-source          inet:ip-prefix
|         |--rw multicast-group           inet:ip-prefix
|     +---:(SHARED_TREE_JOIN)
|         |--rw shared-tree-join-route*
|             [route-type route-distinguisher source-as
|                 multicast-source multicast-group]
|         |--rw route-type                mvpn-route-type
|         |--rw route-distinguisher route-distinguisher-def
|         |--rw source-as                  inet:as-number
|         |--rw multicast-source           inet:ip-prefix
|         |--rw multicast-group            inet:ip-prefix
|     +---:(SOURCE_TREE_JOIN)
|         |--rw source-tree-join-route*
|             [route-type route-distinguisher source-as
|                 multicast-source multicast-group]
|         |--rw route-type                mvpn-route-type
|         |--rw route-distinguisher route-distinguisher-def
|         |--rw source-as                  inet:as-number
|         |--rw multicast-source           inet:ip-prefix
|         |--rw multicast-group            inet:ip-prefix
+--rw bgp-peer-list* [bgp-peer-name]
|   +--rw peer-session-address
|       |--rw local-ipv4-address?      inet:ipv4-prefix
|       |--rw remote-ipv4-address?     inet:ipv4-prefix
+--rw bgp-peer-name                string
+--ro bgp-peer-type?                enumeration
+--ro bgp-peer-create?              enumeration
+--rw bgp-policy-in
+--rw bgp-policy-out
+--ro peer-policy-in-pset
+--ro peer-policy-out-pset
+--rw peer-state-info
|   +--ro peer-current-state?          peer-state
|   +--ro peer-last-state?             peer-state
|   +--ro peer-down-reason

```

```

| | | +--ro error-code?          enumeration
| | | +--ro sub-error-code
| | | |   +--ro (sub-error-code-type)?
| | | |   |   +--:(head-error-sub-code)
| | | |   |   |   +--ro head-error-sub-value? enumeration
| | | |   |   +--:(open-error-sub-code)
| | | |   |   |   +--ro open-error-sub-value? enumeration
| | | |   |   +--:(update-error-sub-code)
| | | |   |   |   +--ro update-error-sub-value?
| | | |   |   |   |   enumeration
| | | |   |   +--:(route-refresh-error-sub-code)
| | | |   |   |   +--ro route-refresh-error-sub-value?
| | | |   |   |   |   enumeration
| | | +--ro peer-received-update-cnt?    uint64
| | | +--ro peer-transmit-update-cnt?    uint64
| | | +--ro peer-received-route-cnt?     uint64
| | | +--ro peer-send-route-cnt?         uint64
| | | +--rw max-prefix-rcv-limit?        uint64
| | | +--rw max-prefix-xmt-limit?        uint64
| | | +--ro peer-prefix-high?            uint64
| | | +--ro peer-prefix-low?             uint64
| | | +--ro peer-perfix-ave?              uint64
| | | +--ro peer-prefix-time?            uint64
| | | +--ro peer-prefix-max?             uint64
| | | +--ro publish-version?             uint64
+--rw bgp-rib-in
| | | +--rw bgp-route-type?              enumeration
| | | +--rw route-admin-distance         uint16
| | | +--rw bgp-attribute-list
| | | |   +--rw bgp-origin?              enumeration
| | | |   +--rw bgp-aspath
| | | |   |   +--rw usascount?            uint16
| | | |   |   +--rw ulrefcount?          uint16
| | | |   |   +--rw asstring?            string
| | | |   |   +--ro usascountfrcomp?     uint16
| | | |   |   +--rw usastotalcount?      uint16
| | | |   |   +--rw usas4pathlen?        uint16
| | | |   |   +--rw as4pathvalue?        string
| | | |   +--rw bgp-nexthop?             inet:ip-address
| | | |   +--rw bgp-med?                  uint32
| | | |   +--rw bgp-localpref?           uint32
| | | |   +--rw bgp-atomic-aggregate?    uint32
| | | |   +--rw bgp-aggregator
| | | |   |   +--rw ulipaddress?          uint32
| | | |   |   +--rw ulasnumber?          uint32
| | | |   +--rw bgp-commattr
| | | |   |   +--rw uscommsize?           uint16
| | | |   |   +--rw ulrefcomm?           uint32

```

```

| | | | +--rw commattr-value?   string
| | | | +--rw bgp-extcommattr
| | | | | +--rw custom-community
| | | | | | +--rw valid           boolean
| | | | | | +--rw insertion-point uint32
| | | | | | +--rw community-id    uint8
| | | | | | +--rw cost-id         uint32
| | | | | +--rw usextcommsize?    uint16
| | | | | +--rw ulrefcount?       uint32
| | | | | +--rw extcommattr-value? string
| | | | +--rw bgp-clusterlist
| | | | | +--rw uscluslen?        uint16
| | | | | +--rw ulrefclus?       uint32
| | | | | +--rw clusterlist-value? string
| | | | +--rw bgp-originator-id? uint32
| | | +--ro bgp-route-create?     enumeration
| | +--ro bgp-rt-state-info
| | | +--ro rib-current-state?    rib-state-def
| | | +--ro rib-last-state?      rib-state-def
| | | +--ro rib-rejected-reason?  enumeration
| | | +--ro not-preferred-reason? enumeration
| | +--rw (bgp-mvpn-route-type)?
| | | +--:(INTRA_AS_I_PMSI_AD)
| | | | +--rw intra-as-i-pmsi-ad-route*
| | | | | [route-type route-distinguisher origin-router-ip-addr]
| | | | | | +--rw route-type          mvpn-route-type
| | | | | | +--rw route-distinguisher route-distinguisher-def
| | | | | | +--rw origin-router-ip-addr inet:ip-prefix
| | | | +--:(INTER_AS_I_PMSI_AD)
| | | | | +--rw inter_as_i_pmsi_as_route*
| | | | | | [route-type route-distinguisher source-as]
| | | | | | | +--rw route-type          mvpn-route-type
| | | | | | | +--rw route-distinguisher route-distinguisher-def
| | | | | | | +--rw source-as           inet:as-number
| | | | +--:(S_PMSI_AD)
| | | | | +--rw s-pmsi-ad-route*
| | | | | | [route-type route-distinguisher
multicast-source multicast-group origin-router-ip-addr]
| | | | | | | +--rw route-type          mvpn-route-type
| | | | | | | +--rw route-distinguisher route-distinguisher-def
| | | | | | | +--rw multicast-source    inet:ip-prefix
| | | | | | | +--rw multicast-group     inet:ip-prefix
| | | | | | | +--rw origin-router-ip-addr inet:ip-prefix
| | | | +--:(LEAF_AD)
| | | | | +--rw leaf-ad-route*
| | | | | | [route-type route-key origin-router-ip-addr]
| | | | | | | +--rw route-type          mvpn-route-type
| | | | | | | +--rw route-key           binary

```

```

|         |      +--rw origin-router-ip-addr      inet:ip-prefix
|         |      +---:(SOURCE_ACTIVE_AD)
|         |          +--rw source-active-ad-route*
[route-type route-distinguisher multicast-source multicast-group]
|         |          +--rw route-type                mvpn-route-type
|         |          +--rw route-distinguisher        route-distinguisher-def
|         |          +--rw multicast-source            inet:ip-prefix
|         |          +--rw multicast-group             inet:ip-prefix
+---:(SHARED_TREE_JOIN)
|         |          +--rw shared-tree-join-route*
[route-type route-distinguisher source-as
|         |              multicast-source multicast-group]
|         |          +--rw route-type                mvpn-route-type
|         |          +--rw route-distinguisher        route-distinguisher-def
|         |          +--rw source-as                  inet:as-number
|         |          +--rw multicast-source            inet:ip-prefix
|         |          +--rw multicast-group             inet:ip-prefix
+---:(SOURCE_TREE_JOIN)
|         |          +--rw source-tree-join-route*
[route-type route-distinguisher source-as
|         |              multicast-source multicast-group]
|         |          +--rw route-type                mvpn-route-type
|         |          +--rw route-distinguisher        route-distinguisher-def
|         |          +--rw source-as                  inet:as-number
|         |          +--rw multicast-source            inet:ip-prefix
|         |          +--rw multicast-group             inet:ip-prefix
+--rw bgp-rib-out
|     +--rw bgp-route-type?           enumeration
|     +--rw route-admin-distance       uint16
|     +--rw bgp-attribute-list
|         |     +--rw bgp-origin?       enumeration
|         |     +--rw bgp-aspath
|         |         |     +--rw usascount?    uint16
|         |         |     +--rw ulrefcount?   uint16
|         |         |     +--rw asstring?     string
|         |         |     +--ro usascountfrcomp? uint16
|         |         |     +--rw usastotalcount? uint16
|         |         |     +--rw usas4pathlen? uint16
|         |         |     +--rw as4pathvalue? string
|         |     +--rw bgp-nexthop?         inet:ip-address
|         |     +--rw bgp-med?              uint32
|         |     +--rw bgp-localpref?        uint32
|         |     +--rw bgp-atomic-aggregate? uint32
|         |     +--rw bgp-aggregator
|         |         |     +--rw ulipaddress?   uint32
|         |         |     +--rw ulasnumber?    uint32
+--rw bgp-commattr
|     +--rw uscommsize?                 uint16

```



```

| | +--rw ulrefcomm?          uint32
| | +--rw commattr-value?    string
+--rw bgp-extcommattr
| | +--rw custom-community
| | | +--rw valid            boolean
| | | +--rw insertion-point  uint32
| | | +--rw community-id     uint8
| | | +--rw cost-id          uint32
| | +--rw usextcommsize?     uint16
| | +--rw ulrefcount?        uint32
| | +--rw extcommattr-value? string
+--rw bgp-clusterlist
| | +--rw uscluslen?         uint16
| | +--rw ulrefclus?         uint32
| | +--rw clusterlist-value? string
+--rw bgp-originator-id?     uint32
+--ro bgp-route-create?      enumeration
+--ro bgp-rt-state-info
| | +--ro rib-current-state?  rib-state-def
| | +--ro rib-last-state?     rib-state-def
| | +--ro rib-rejected-reason? enumeration
| | +--ro not-preferred-reason? enumeration
+--rw (bgp-mvpn-route-type)?
| | +--:(INTRA_AS_I_PMSI_AD)
| | | +--rw intra-as-i-pmsi-ad-route*
[route-type route-distinguisher origin-router-ip-addr]
| | | +--rw route-type           mvpn-route-type
| | | +--rw route-distinguisher route-distinguisher-def
| | | +--rw origin-router-ip-addr inet:ip-prefix
+--:(INTER_AS_I_PMSI_AD)
| | +--rw inter_as_i_pmsi_as_route*
| | | [route-type route-distinguisher source-as]
| | | +--rw route-type           mvpn-route-type
| | | +--rw route-distinguisher route-distinguisher-def
| | | +--rw source-as             inet:as-number
+--:(S_PMSI_AD)
| | +--rw s-pmsi-ad-route*
| | | [route-type route-distinguisher multicast-source
| | |   multicast-group origin-router-ip-addr]
| | | +--rw route-type           mvpn-route-type
| | | +--rw route-distinguisher route-distinguisher-def
| | | +--rw multicast-source      inet:ip-prefix
| | | +--rw multicast-group       inet:ip-prefix
| | | +--rw origin-router-ip-addr inet:ip-prefix
+--:(LEAF_AD)
| | +--rw leaf-ad-route*
| | | [route-type route-key origin-router-ip-addr]
| | | +--rw route-type           mvpn-route-type

```

```

|         |         |---rw route-key                binary
|         |         |--rw origin-router-ip-addr    inet:ip-prefix
+---:(SOURCE_ACTIVE_AD)
|         |--rw source-active-ad-route*
[route-type route-distinguisher multicast-source multicast-group]
|         |--rw route-type                    mvpn-route-type
|         |--rw route-distinguisher route-distinguisher-def
|         |--rw multicast-source              inet:ip-prefix
|         |--rw multicast-group              inet:ip-prefix
+---:(SHARED_TREE_JOIN)
|         |--rw shared-tree-join-route*
|         [route-type route-distinguisher source-as
|           multicast-source multicast-group]
|         |--rw route-type                    mvpn-route-type
|         |--rw route-distinguisher route-distinguisher-def
|         |--rw source-as                     inet:as-number
|         |--rw multicast-source             inet:ip-prefix
|         |--rw multicast-group             inet:ip-prefix
+---:(SOURCE_TREE_JOIN)
|         |--rw source-tree-join-route*
|         [route-type route-distinguisher source-as
|           multicast-source multicast-group]
|         |--rw route-type                    mvpn-route-type
|         |--rw route-distinguisher route-distinguisher-def
|         |--rw source-as                     inet:as-number
|         |--rw multicast-source             inet:ip-prefix
|         |--rw multicast-group             inet:ip-prefix
+--rw bgp-ipv6-mvpn-instance
|   +--rw bgp-instance-name?                  string
|   +--rw bgp-instance-create?               enumeration
|   +--rw bgp-instance-type?                 enumeration
|   +--rw bgp-vendor-type?                   enumeration
|   +--rw afi?                               afi-def
|   +--rw safi?                              safi-def
+--rw bgp-local-rib
|   +--rw bgp-route-type?                    enumeration
|   +--rw route-admin-distance uint16
|   +--rw bgp-attribute-list
|     +--rw bgp-origin?                      enumeration
|     +--rw bgp-aspath
|       +--rw usascalcount?                  uint16
|       +--rw ulrefcount?                   uint16
|       +--rw asstring?                     string
|       +--ro usascalcountfrcomp?          uint16
|       +--rw usastotalcount?              uint16
|       +--rw asas4pathlen?                uint16
|       +--rw asas4pathvalue?              string
+--rw bgp-nexthop?                          inet:ip-address

```

```

+--rw bgp-med?                               uint32
+--rw bgp-localpref?                         uint32
+--rw bgp-atomic-aggregate?                 uint32
+--rw bgp-aggregator
|   +--rw ulipaddress?                       uint32
|   +--rw ulasnumber?                       uint32
+--rw bgp-commattr
|   +--rw uscommsize?                       uint16
|   +--rw ulrefcomm?                       uint32
|   +--rw commattr-value?                   string
+--rw bgp-extcommattr
|   +--rw custom-community
|   |   +--rw valid                         boolean
|   |   +--rw insertion-point              uint32
|   |   +--rw community-id                 uint8
|   |   +--rw cost-id                      uint32
|   +--rw usextcommsize?                   uint16
|   +--rw ulrefcount?                     uint32
|   +--rw extcommattr-value?               string
+--rw bgp-clusterlist
|   +--rw uscluslen?                       uint16
|   +--rw ulrefclus?                      uint32
|   +--rw clusterlist-value?               string
+--rw bgp-originator-id?                   uint32
+--ro bgp-route-create?                     enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state?                rib-state-def
|   +--ro rib-last-state?                  rib-state-def
|   +--ro rib-rejected-reason?              enumeration
|   +--ro not-preferred-reason?             enumeration
+--rw (bgp-mvpn-route-type)?
+--:( INTRA_AS_I_PMSI_AD)
|   +--rw intra-as-i-pmsi-ad-route*
|   |   [route-type route-distinguisher origin-router-ip-addr]
|   |   +--rw route-type                   mvpn-route-type
|   |   +--rw route-distinguisher           route-distinguisher-def
|   |   +--rw origin-router-ip-addr         inet:ip-prefix
+--:( INTER_AS_I_PMSI_AD)
|   +--rw inter_as_i_pmsi_as_route*
|   |   [route-type route-distinguisher source-as]
|   |   +--rw route-type                   mvpn-route-type
|   |   +--rw route-distinguisher           route-distinguisher-def
|   |   +--rw source-as                     inet:as-number
+--:( S_PMSI_AD)
|   +--rw s-pmsi-ad-route*
|   |   [route-type route-distinguisher multicast-source
|   |   |   multicast-group origin-router-ip-addr]
|   |   +--rw route-type                   mvpn-route-type

```

```

|         +---rw route-distinguisher route-distinguisher-def
|         +---rw multicast-source      inet:ip-prefix
|         +---rw multicast-group       inet:ip-prefix
|         +---rw origin-router-ip-addr inet:ip-prefix
+---:(LEAF_AD)
|   +---rw leaf-ad-route*
|       [route-type route-key origin-router-ip-addr]
|       +---rw route-type      mvpn-route-type
|       +---rw route-key      binary
|       +---rw origin-router-ip-addr  inet:ip-prefix
+---:(SOURCE_ACTIVE_AD)
|   +---rw source-active-ad-route*
|       [route-type route-distinguisher
|         multicast-source multicast-group]
|       +---rw route-type      mvpn-route-type
|       +---rw route-distinguisher route-distinguisher-def
|       +---rw multicast-source  inet:ip-prefix
|       +---rw multicast-group   inet:ip-prefix
+---:(SHARED_TREE_JOIN)
|   +---rw shared-tree-join-route*
|       [route-type route-distinguisher
|         source-as multicast-source multicast-group]
|       +---rw route-type      mvpn-route-type
|       +---rw route-distinguisher route-distinguisher-def
|       +---rw source-as      inet:as-number
|       +---rw multicast-source inet:ip-prefix
|       +---rw multicast-group inet:ip-prefix
+---:(SOURCE_TREE_JOIN)
|   +---rw source-tree-join-route*
|       [route-type route-distinguisher source-as
|         multicast-source multicast-group]
|       +---rw route-type      mvpn-route-type
|       +---rw route-distinguisher route-distinguisher-def
|       +---rw source-as      inet:as-number
|       +---rw multicast-source inet:ip-prefix
|       +---rw multicast-group inet:ip-prefix
+---rw bgp-peer-list* [bgp-peer-name]
|   +---rw peer-session-address
|       |   +---rw local-ipv6-address?  inet:ipv6-prefix
|       |   +---rw remote-ipv6-address? inet:ipv6-prefix
+---rw bgp-peer-name      string
+---ro bgp-peer-type?     enumeration
+---ro bgp-peer-create?   enumeration
+---rw bgp-policy-in
+---rw bgp-policy-out
+---ro peer-policy-in-pset
+---ro peer-policy-out-pset
+---rw peer-state-info

```

```

+--ro peer-current-state?          peer-state
+--ro peer-last-state?             peer-state
+--ro peer-down-reason
|   +--ro error-code?              enumeration
|   +--ro sub-error-code
|       +--ro (sub-error-code-type)?
|           +--:(head-error-sub-code)
|               |   +--ro head-error-sub-value?
|                   enumeration
|               +--:(open-error-sub-code)
|                   |   +--ro open-error-sub-value?
|                       enumeration
|               +--:(update-error-sub-code)
|                   |   +--ro update-error-sub-value?
|                       enumeration
|               +--:(route-refresh-error-sub-code)
|                   +--ro route-refresh-error-sub-value?
|                       enumeration
+--ro peer-received-update-cnt?    uint64
+--ro peer-transmit-update-cnt?    uint64
+--ro peer-received-route-cnt?    uint64
+--ro peer-send-route-cnt?        uint64
+--rw max-prefix-rcv-limit?       uint64
+--rw max-prefix-xmt-limit?       uint64
+--ro peer-prefix-high?           uint64
+--ro peer-prefix-low?            uint64
+--ro peer-perfix-ave?             uint64
+--ro peer-prefix-time?           uint64
+--ro peer-prefix-max?            uint64
+--ro publish-version?            uint64
+--rw bgp-rib-in
|   +--rw bgp-route-type?          enumeration
|   +--rw route-admin-distance    uint16
|   +--rw bgp-attribute-list
|       +--rw bgp-origin?          enumeration
|       +--rw bgp-aspath
|           +--rw usascount?        uint16
|           +--rw ulrefcount?       uint16
|           +--rw asstring?         string
|           +--ro usascountfrcomp?  uint16
|           +--rw usastotalcount?   uint16
|           +--rw usas4pathlen?     uint16
|           +--rw as4pathvalue?     string
|   +--rw bgp-nexthop?            inet:ip-address
|   +--rw bgp-med?                 uint32
|   +--rw bgp-localpref?           uint32
|   +--rw bgp-atomic-aggregate?    uint32
|   +--rw bgp-aggregator

```

```

|--rw ulipaddress?      uint32
+--rw ulasnumber?      uint32
+--rw bgp-commattr
  +--rw uscommsize?      uint16
  +--rw ulrefcomm?      uint32
  +--rw commattr-value?  string
+--rw bgp-extcommattr
  +--rw custom-community
  +--rw valid            boolean
  +--rw insertion-point  uint32
  +--rw community-id     uint8
  +--rw cost-id          uint32
  +--rw usextcommsize?   uint16
  +--rw ulrefcount?      uint32
  +--rw extcommattr-value? string
+--rw bgp-clusterlist
  +--rw uscluslen?      uint16
  +--rw ulrefclus?      uint32
  +--rw clusterlist-value? string
+--rw bgp-originator-id? uint32
+--ro bgp-route-create?  enumeration
+--ro bgp-rt-state-info
  +--ro rib-current-state?  rib-state-def
  +--ro rib-last-state?     rib-state-def
  +--ro rib-rejected-reason? enumeration
  +--ro not-preferred-reason? enumeration
+--rw (bgp-mvpn-route-type)?
  +--:( INTRA_AS_I_PMSI_AD)
    +--rw intra-as-i-pmsi-ad-route*
[route-type route-distinguisher origin-router-ip-addr]
  +--rw route-type          mvpn-route-type
  +--rw route-distinguisher route-distinguisher-def
  +--rw origin-router-ip-addr  inet:ip-prefix
+--:( INTER_AS_I_PMSI_AD)
  +--rw inter_as_i_pmsi_as_route*
[route-type route-distinguisher source-as]
  +--rw route-type          mvpn-route-type
  +--rw route-distinguisher route-distinguisher-def
  +--rw source-as           inet:as-number
+--:( S_PMSI_AD)
  +--rw s-pmsi-ad-route*
    [route-type route-distinguisher multicast-source
      multicast-group origin-router-ip-addr]
      +--rw route-type          mvpn-route-type
      +--rw route-distinguisher route-distinguisher-def
      +--rw multicast-source     inet:ip-prefix
      +--rw multicast-group      inet:ip-prefix
      +--rw origin-router-ip-addr inet:ip-prefix

```

```

+---:(LEAF_AD)
|   +---rw leaf-ad-route*
|   [route-type route-key origin-router-ip-addr]
|       +---rw route-type                mvpn-route-type
|       +---rw route-key                  binary
|       +---rw origin-router-ip-addr      inet:ip-prefix
+---:(SOURCE_ACTIVE_AD)
|   +---rw source-active-ad-route*
|       [route-type route-distinguisher
|           multicast-source multicast-group]
|       +---rw route-type                mvpn-route-type
|       +---rw route-distinguisher route-distinguisher-def
|       +---rw multicast-source          inet:ip-prefix
|       +---rw multicast-group           inet:ip-prefix
+---:(SHARED_TREE_JOIN)
|   +---rw shared-tree-join-route*
|       [route-type route-distinguisher source-as
|           multicast-source multicast-group]
|       +---rw route-type                mvpn-route-type
|       +---rw route-distinguisher route-distinguisher-def
|       +---rw source-as                  inet:as-number
|       +---rw multicast-source           inet:ip-prefix
|       +---rw multicast-group            inet:ip-prefix
+---:(SOURCE_TREE_JOIN)
|   +---rw source-tree-join-route*
|       [route-type route-distinguisher source-as
|           multicast-source multicast-group]
|       +---rw route-type                mvpn-route-type
|       +---rw route-distinguisher route-distinguisher-def
|       +---rw source-as                  inet:as-number
|       +---rw multicast-source           inet:ip-prefix
|       +---rw multicast-group            inet:ip-prefix
+---rw bgp-rib-out
|   +---rw bgp-route-type?                enumeration
|   +---rw route-admin-distance          uint16
|   +---rw bgp-attribute-list
|       +---rw bgp-origin?                enumeration
|       +---rw bgp-aspath
|           +---rw usascount?              uint16
|           +---rw ulrefcount?             uint16
|           +---rw asstring?               string
|           +---ro usascountfrcomp?        uint16
|           +---rw usastotalcount?         uint16
|           +---rw usas4pathlen?           uint16
|           +---rw as4pathvalue?           string
|       +---rw bgp-nexthop?                inet:ip-address
|       +---rw bgp-med?                    uint32
|       +---rw bgp-localpref?              uint32

```

```

+--rw bgp-atomic-aggregate?  uint32
+--rw bgp-aggregator
|   +--rw ulipaddress?  uint32
|   +--rw ulasnumber?   uint32
+--rw bgp-commattr
|   +--rw uscommsize?    uint16
|   +--rw ulrefcomm?    uint32
|   +--rw commattr-value? string
+--rw bgp-extcommattr
|   +--rw custom-community
|   |   +--rw valid      boolean
|   |   +--rw insertion-point uint32
|   |   +--rw community-id  uint8
|   |   +--rw cost-id      uint32
|   +--rw usextcommsize?  uint16
|   +--rw ulrefcount?    uint32
|   +--rw extcommattr-value? string
+--rw bgp-clusterlist
|   +--rw uscluslen?      uint16
|   +--rw ulrefclus?     uint32
|   +--rw clusterlist-value? string
+--rw bgp-originator-id?   uint32
+--ro bgp-route-create?    enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state?  rib-state-def
|   +--ro rib-last-state?    rib-state-def
|   +--ro rib-rejected-reason? enumeration
|   +--ro not-preferred-reason? enumeration
+--rw (bgp-mvpn-route-type)?
+--:( INTRA_AS_I_PMSI_AD)
|   +--rw intra-as-i-pmsi-ad-route*
|   [route-type route-distinguisher origin-router-ip-addr]
|   |   +--rw route-type          mvpn-route-type
|   |   +--rw route-distinguisher route-distinguisher-def
|   |   +--rw origin-router-ip-addr  inet:ip-prefix
+--:( INTER_AS_I_PMSI_AD)
|   +--rw inter_as_i_pmsi_as_route*
|   |   [route-type route-distinguisher source-as]
|   |   +--rw route-type          mvpn-route-type
|   |   +--rw route-distinguisher route-distinguisher-def
|   |   +--rw source-as            inet:as-number
+--:( S_PMSI_AD)
|   +--rw s-pmsi-ad-route*
|   |   [route-type route-distinguisher multicast-source
|   |   |   multicast-group origin-router-ip-addr]
|   |   +--rw route-type          mvpn-route-type
|   |   +--rw route-distinguisher route-distinguisher-def
|   |   +--rw multicast-source     inet:ip-prefix

```



```

|         |--rw multicast-group          inet:ip-prefix
|         |--rw origin-router-ip-addr    inet:ip-prefix
|---:(LEAF_AD)
|         |--rw leaf-ad-route*
|         |         [route-type route-key origin-router-ip-addr]
|         |--rw route-type              mvpn-route-type
|         |--rw route-key                binary
|         |--rw origin-router-ip-addr    inet:ip-prefix
|---:(SOURCE_ACTIVE_AD)
|         |--rw source-active-ad-route*
|         |         [route-type route-distinguisher
|         |         |         multicast-source multicast-group]
|         |--rw route-type              mvpn-route-type
|         |--rw route-distinguisher      route-distinguisher-def
|         |--rw multicast-source          inet:ip-prefix
|         |--rw multicast-group          inet:ip-prefix
|---:(SHARED_TREE_JOIN)
|         |--rw shared-tree-join-route*
|         |         [route-type route-distinguisher source-as
|         |         |         multicast-source multicast-group]
|         |--rw route-type              mvpn-route-type
|         |--rw route-distinguisher      route-distinguisher-def
|         |--rw source-as                inet:as-number
|         |--rw multicast-source          inet:ip-prefix
|         |--rw multicast-group          inet:ip-prefix
|---:(SOURCE_TREE_JOIN)
|         |--rw source-tree-join-route*
|         |         [route-type route-distinguisher source-as
|         |         |         multicast-source multicast-group]
|         |--rw route-type              mvpn-route-type
|         |--rw route-distinguisher      route-distinguisher-def
|         |--rw source-as                inet:as-number
|         |--rw multicast-source          inet:ip-prefix
|         |--rw multicast-group          inet:ip-prefix
+--rw bgp-mcast-vpls-instance
|   |--rw bgp-instance-name?            string
|   |--rw bgp-instance-create?          enumeration
|   |--rw bgp-instance-type?            enumeration
|   |--rw bgp-vendor-type?              enumeration
|   |--rw afi?                          afi-def
|   |--rw safi?                          safi-def
|   |--rw bgp-local-rib
|   |   |--rw bgp-route-type?            enumeration
|   |   |--rw route-admin-distance      uint16
|   |   |--rw bgp-attribute-list
|   |   |   |--rw bgp-origin?            enumeration
|   |   |   |--rw bgp-aspath
|   |   |   |--rw usascount?             uint16

```

```

+--rw ulrefcount?          uint16
+--rw asstring?            string
+--ro usascountfrcomp?     uint16
+--rw usastotalcount?      uint16
+--rw usas4pathlen?        uint16
+--rw as4pathvalue?        string
+--rw bgp-nexthop?         inet:ip-address
+--rw bgp-med?             uint32
+--rw bgp-localpref?       uint32
+--rw bgp-atomic-aggregate? uint32
+--rw bgp-aggregator
|   +--rw ulipaddress?      uint32
|   +--rw ulasnumber?       uint32
+--rw bgp-commattr
|   +--rw uscommsize?       uint16
|   +--rw ulrefcomm?        uint32
|   +--rw commattr-value?   string
+--rw bgp-extcommattr
|   +--rw custom-community
|   |   +--rw valid         boolean
|   |   +--rw insertion-point uint32
|   |   +--rw community-id  uint8
|   |   +--rw cost-id       uint32
|   +--rw usextcommsize?    uint16
|   +--rw ulrefcount?       uint32
|   +--rw extcommattr-value? string
+--rw bgp-clusterlist
|   +--rw uscluslen?        uint16
|   +--rw ulrefclus?        uint32
|   +--rw clusterlist-value? string
+--rw bgp-originator-id?   uint32
+--ro bgp-route-create?     enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state? rib-state-def
|   +--ro rib-last-state?   rib-state-def
|   +--ro rib-rejected-reason? enumeration
|   +--ro not-preferred-reason? enumeration
+--rw (bgp-mcast-vpls-route-type)?
+--:(SELECTIVE_TREE_AD_ROUTE)
|   +--rw selective-tree-ad-route
[route-type route-distinguisher multicast-source
    multicast-group origin-router-ip-addr]
|   +--rw route-type        mcast-vpls-route-type
|   +--rw route-distinguisher
|   |   route-distinguisher-def
|   |   +--rw multicast-source    inet:ip-prefix
|   |   +--rw multicast-group     inet:ip-prefix
|   |   +--rw origin-router-ip-addr inet:ip-prefix

```

```

| |      +---:(LEAF_AD)
| |      +---rw leaf-ad-route*
| |          [route-type route-key origin-router-ip-addr]
| |      +---rw route-type          mcast-vpls-route-type
| |      +---rw route-key          binary
| |      +---rw origin-router-ip-addr    inet:ip-prefix
+---rw bgp-peer-list* [bgp-peer-name]
+---rw peer-session-address
|   +---rw local-ipv4-address?    inet:ipv4-prefix
|   +---rw remote-ipv4-address?   inet:ipv4-prefix
+---rw bgp-peer-name              string
+---ro bgp-peer-type?              enumeration
+---ro bgp-peer-create?            enumeration
+---rw bgp-policy-in
+---rw bgp-policy-out
+---ro peer-policy-in-pset
+---ro peer-policy-out-pset
+---rw peer-state-info
|   +---ro peer-current-state?      peer-state
|   +---ro peer-last-state?         peer-state
|   +---ro peer-down-reason
|   |   +---ro error-code?          enumeration
|   |   +---ro sub-error-code
|   |       +---ro (sub-error-code-type)?
|   |       +---:(head-error-sub-code)
|   |       |   +---ro head-error-sub-value?
|   |       |       enumeration
|   |       +---:(open-error-sub-code)
|   |       |   +---ro open-error-sub-value?
|   |       |       enumeration
|   |       +---:(update-error-sub-code)
|   |       |   +---ro update-error-sub-value?
|   |       |       enumeration
|   |       +---:(route-refresh-error-sub-code)
|   |       |   +---ro route-refresh-error-sub-value?
|   |       |       enumeration
+---ro peer-received-update-cnt?    uint64
+---ro peer-transmit-update-cnt?    uint64
+---ro peer-received-route-cnt?     uint64
+---ro peer-send-route-cnt?         uint64
+---rw max-prefix-rcv-limit?        uint64
+---rw max-prefix-xmt-limit?        uint64
+---ro peer-prefix-high?            uint64
+---ro peer-prefix-low?             uint64
+---ro peer-perfix-ave?              uint64
+---ro peer-prefix-time?            uint64
+---ro peer-prefix-max?             uint64
+---ro publish-version?             uint64

```

```

+--rw bgp-rib-in
|   +--rw bgp-route-type?          enumeration
|   +--rw route-admin-distance    uint16
|   +--rw bgp-attribute-list
|   |   +--rw bgp-origin?          enumeration
|   |   +--rw bgp-aspath
|   |   |   +--rw usascount?        uint16
|   |   |   +--rw ulrefcount?      uint16
|   |   |   +--rw asstring?        string
|   |   |   +--ro usascountfrcomp?  uint16
|   |   |   +--rw usastotalcount?  uint16
|   |   |   +--rw usas4pathlen?    uint16
|   |   |   +--rw as4pathvalue?    string
|   |   +--rw bgp-nexthop?        inet:ip-address
|   |   +--rw bgp-med?             uint32
|   |   +--rw bgp-localpref?       uint32
|   |   +--rw bgp-atomic-aggregate? uint32
|   |   +--rw bgp-aggregator
|   |   |   +--rw ulipaddress?      uint32
|   |   |   +--rw ulasnumber?      uint32
|   |   +--rw bgp-commattr
|   |   |   +--rw uscommsize?        uint16
|   |   |   +--rw ulrefcomm?        uint32
|   |   |   +--rw commattr-value?   string
|   |   +--rw bgp-extcommattr
|   |   |   +--rw custom-community
|   |   |   |   +--rw valid          boolean
|   |   |   |   +--rw insertion-point uint32
|   |   |   |   +--rw community-id   uint8
|   |   |   |   +--rw cost-id        uint32
|   |   |   +--rw usextcommsize?    uint16
|   |   |   +--rw ulrefcount?       uint32
|   |   |   +--rw extcommattr-value? string
|   |   +--rw bgp-clusterlist
|   |   |   +--rw uscluslen?        uint16
|   |   |   +--rw ulrefclus?       uint32
|   |   |   +--rw clusterlist-value? string
|   |   +--rw bgp-originator-id?   uint32
|   +--ro bgp-route-create?         enumeration
|   +--ro bgp-rt-state-info
|   |   +--ro rib-current-state?    rib-state-def
|   |   +--ro rib-last-state?       rib-state-def
|   |   +--ro rib-rejected-reason?  enumeration
|   |   +--ro not-preferred-reason?  enumeration
|   +--rw (bgp-mcast-vpls-route-type)?
|   |   +--:(SELECTIVE_TREE_AD_ROUTE)
|   |   |   +--rw selective-tree-ad-route*
|   |   |   [route-type route-distinguisher multicast-source

```

```

        multicast-group origin-router-ip-addr]
+--rw route-type          mcast-vpls-route-type
+--rw route-distinguisher
    route-distinguisher-def
+--rw multicast-source      inet:ip-prefix
+--rw multicast-group       inet:ip-prefix
+--rw origin-router-ip-addr inet:ip-prefix
+--:(LEAF_AD)
+--rw leaf-ad-route*
    [route-type route-key origin-router-ip-addr]
+--rw route-type          mcast-vpls-route-type
+--rw route-key            binary
+--rw origin-router-ip-addr inet:ip-prefix
+--rw bgp-rib-out
+--rw bgp-route-type?      enumeration
+--rw route-admin-distance uint16
+--rw bgp-attribute-list
    +--rw bgp-origin?       enumeration
    +--rw bgp-aspath
        +--rw usascount?    uint16
        +--rw ulrefcount?   uint16
        +--rw asstring?     string
        +--ro usascountfrcomp? uint16
        +--rw usastotalcount? uint16
        +--rw usas4pathlen? uint16
        +--rw as4pathvalue? string
    +--rw bgp-nexthop?      inet:ip-address
    +--rw bgp-med?          uint32
    +--rw bgp-localpref?    uint32
    +--rw bgp-atomic-aggregate? uint32
    +--rw bgp-aggregator
        +--rw ulipaddress?   uint32
        +--rw ulasnumber?    uint32
    +--rw bgp-commattr
        +--rw uscommsize?    uint16
        +--rw ulrefcomm?     uint32
        +--rw commattr-value? string
    +--rw bgp-extcommattr
        +--rw custom-community
            +--rw valid      boolean
            +--rw insertion-point uint32
            +--rw community-id  uint8
            +--rw cost-id      uint32
        +--rw usextcommsize?   uint16
        +--rw ulrefcount?      uint32
        +--rw extcommattr-value? string
+--rw bgp-clusterlist
    +--rw uscluslen?          uint16

```

```

|         |         |   +--rw ulrefclus?           uint32
|         |         |   +--rw clusterlist-value?    string
|         |         |   +--rw bgp-originator-id?    uint32
+--ro bgp-route-create?           enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state?      rib-state-def
|   +--ro rib-last-state?         rib-state-def
|   +--ro rib-rejected-reason?    enumeration
|   +--ro not-preferred-reason?   enumeration
+--rw (bgp-mcast-vpls-route-type)?
|   +--:(SELECTIVE_TREE_AD_ROUTE)
|   |   +--rw selective-tree-ad-route*
|   |   [route-type route-distinguisher multicast-source
|   |       multicast-group origin-router-ip-addr]
|   |   +--rw route-type          mcast-vpls-route-type
|   |   +--rw route-distinguisher route-distinguisher-def
|   |   +--rw multicast-source     inet:ip-prefix
|   |   +--rw multicast-group      inet:ip-prefix
|   |   +--rw origin-router-ip-addr inet:ip-prefix
|   +--:(LEAF_AD)
|   |   +--rw leaf-ad-route*
|   |   [route-type route-key origin-router-ip-addr]
|   |   +--rw route-type          mcast-vpls-route-type
|   |   +--rw route-key           binary
|   |   +--rw origin-router-ip-addr inet:ip-prefix
+--rw bgp-evpn-instance
|   +--rw bgp-instance-name?      string
|   +--rw bgp-instance-create?    enumeration
|   +--rw bgp-instance-type?      enumeration
|   +--rw bgp-vendor-type?        enumeration
|   +--rw afi?                    afi-def
|   +--rw safi?                   safi-def
+--rw bgp-local-rib
|   +--rw bgp-route-type?         enumeration
|   +--rw route-admin-distance    uint16
+--rw bgp-attribute-list
|   +--rw bgp-origin?             enumeration
+--rw bgp-aspath
|   +--rw usascount?              uint16
|   +--rw ulrefcount?             uint16
|   +--rw asstring?               string
|   +--ro usascountfrcomp?        uint16
|   +--rw usastotalcount?         uint16
|   +--rw usas4pathlen?           uint16
|   +--rw as4pathvalue?           string
+--rw bgp-nexthop?                inet:ip-address
+--rw bgp-med?                    uint32
+--rw bgp-localpref?              uint32

```

```

|--rw bgp-atomic-aggregate?      uint32
+--rw bgp-aggregator
|   +--rw ulipaddress?           uint32
|   +--rw ulasnumber?           uint32
+--rw bgp-commattr
|   +--rw uscommsize?            uint16
|   +--rw ulrefcomm?            uint32
|   +--rw commattr-value?       string
+--rw bgp-extcommattr
|   +--rw custom-community
|   |   +--rw valid              boolean
|   |   +--rw insertion-point   uint32
|   |   +--rw community-id      uint8
|   |   +--rw cost-id           uint32
|   +--rw usextcommsize?        uint16
|   +--rw ulrefcount?           uint32
|   +--rw extcommattr-value?    string
+--rw bgp-clusterlist
|   +--rw uscluslen?            uint16
|   +--rw ulrefclus?           uint32
|   +--rw clusterlist-value?    string
+--rw bgp-originator-id?        uint32
+--ro bgp-route-create?          enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state?     rib-state-def
|   +--ro rib-last-state?       rib-state-def
|   +--ro rib-rejected-reason?   enumeration
|   +--ro not-preferred-reason?  enumeration
+--rw (bgp-evpn-route-type)?
|   +--:(EVPN_AD_ROUTE)
|   |   +--rw evpn-ad-route*
|   |   [route-type route-distinguisher esi eth-tag-id]
|   |   |   +--rw route-type          evpn-route-type
|   |   |   +--rw route-distinguisher route-distinguisher-def
|   |   |   +--rw esi                  string
|   |   |   +--rw eth-tag-id           uint32
|   |   +--:(EVPN_MAC_ROUTE)
|   |   |   +--rw evpn-mac-route*
|   |   |   [route-type route-distinguisher esi eth-tag-id
|   |   |   mac-length mac-address ip-address-length ip-address]
|   |   |   |   +--rw route-type          evpn-route-type
|   |   |   |   +--rw route-distinguisher route-distinguisher-def
|   |   |   |   +--rw esi                  string
|   |   |   |   +--rw eth-tag-id           uint32
|   |   |   |   +--rw mac-length          uint8
|   |   |   |   +--rw mac-address         yang:mac-address
|   |   |   |   +--rw ip-address-length   uint8
|   |   |   |   +--rw ip-address          inet:ip-address

```

```

|         |         +---rw label1?          uint32
|         |         +---rw label2?          uint32
|         +---:(EVPN_INC_MCAST_TREE)
|         |         +---rw evpn-inc-mcast-tree-route*
|         |         |         [route-type route-distinguisher eth-tag-id
|         |         |         |         ip-address-length origin-router-ip-addr]
|         |         |         +---rw route-type          evpn-route-type
|         |         |         +---rw route-distinguisher route-distinguisher-def
|         |         |         +---rw eth-tag-id          uint32
|         |         |         +---rw ip-address-length   uint8
|         |         |         +---rw origin-router-ip-addr inet:ip-prefix
|         +---:(EVPN_ETH_SEGMENT_ROUTE)
|         |         +---rw evpn-eth-segment-route*
|         |         |         [route-type route-distinguisher esi
|         |         |         |         ip-address-length origin-router-ip-addr]
|         |         |         +---rw route-type          evpn-route-type
|         |         |         +---rw route-distinguisher route-distinguisher-def
|         |         |         +---rw esi                  string
|         |         |         +---rw ip-address-length   uint8
|         |         |         +---rw origin-router-ip-addr inet:ip-prefix
+---rw bgp-peer-list* [bgp-peer-name]
+---rw peer-session-address
|   +---rw local-ipv4-address?  inet:ipv4-prefix
|   +---rw remote-ipv4-address? inet:ipv4-prefix
+---rw bgp-peer-name            string
+---ro bgp-peer-type?           enumeration
+---ro bgp-peer-create?        enumeration
+---rw bgp-policy-in
+---rw bgp-policy-out
+---ro peer-policy-in-pset
+---ro peer-policy-out-pset
+---rw peer-state-info
|   +---ro peer-current-state?   peer-state
|   +---ro peer-last-state?     peer-state
|   +---ro peer-down-reason
|   |   +---ro error-code?       enumeration
|   |   +---ro sub-error-code
|   |   |   +---ro (sub-error-code-type)?
|   |   |   |   +---:(head-error-sub-code)
|   |   |   |   |   +---ro head-error-sub-value? enumeration
|   |   |   |   |   +---:(open-error-sub-code)
|   |   |   |   |   |   +---ro open-error-sub-value? enumeration
|   |   |   |   |   |   +---:(update-error-sub-code)
|   |   |   |   |   |   |   +---ro update-error-sub-value? enumeration
|   |   |   |   |   |   |   +---:(route-refresh-error-sub-code)
|   |   |   |   |   |   |   |   +---ro route-refresh-error-sub-value?
|   |   |   |   |   |   |   |   |   enumeration
|   +---ro peer-received-update-cnt?  uint64

```



```

|   +---ro peer-transmit-update-cnt?   uint64
|   +---ro peer-received-route-cnt?   uint64
|   +---ro peer-send-route-cnt?       uint64
|   +---rw max-prefix-rcv-limit?      uint64
|   +---rw max-prefix-xmt-limit?      uint64
|   +---ro peer-prefix-high?          uint64
|   +---ro peer-prefix-low?           uint64
|   +---ro peer-perfix-ave?            uint64
|   +---ro peer-prefix-time?          uint64
|   +---ro peer-prefix-max?           uint64
|   +---ro publish-version?           uint64
+---rw bgp-rib-in
|   +---rw bgp-route-type?             enumeration
|   +---rw route-admin-distance        uint16
|   +---rw bgp-attribute-list
|   |   +---rw bgp-origin?             enumeration
|   |   +---rw bgp-aspath
|   |   |   +---rw usascount?           uint16
|   |   |   +---rw ulrefcount?         uint16
|   |   |   +---rw asstring?           string
|   |   |   +---ro usascountfrcomp?    uint16
|   |   |   +---rw usastotalcount?     uint16
|   |   |   +---rw usas4pathlen?       uint16
|   |   |   +---rw as4pathvalue?       string
|   |   +---rw bgp-nexthop?            inet:ip-address
|   |   +---rw bgp-med?                 uint32
|   |   +---rw bgp-localpref?           uint32
|   |   +---rw bgp-atomic-aggregate?    uint32
|   |   +---rw bgp-aggregator
|   |   |   +---rw ulipaddress?         uint32
|   |   |   +---rw ulasnumber?         uint32
|   |   +---rw bgp-commattr
|   |   |   +---rw uscommsize?          uint16
|   |   |   +---rw ulrefcomm?          uint32
|   |   |   +---rw commattr-value?     string
|   |   +---rw bgp-extcommattr
|   |   |   +---rw custom-community
|   |   |   |   +---rw valid            boolean
|   |   |   |   +---rw insertion-point uint32
|   |   |   |   +---rw community-id    uint8
|   |   |   |   +---rw cost-id         uint32
|   |   |   +---rw usextcommsize?      uint16
|   |   |   +---rw ulrefcount?         uint32
|   |   |   +---rw extcommattr-value?  string
|   |   +---rw bgp-clusterlist
|   |   |   +---rw uscluslen?           uint16
|   |   |   +---rw ulrefclus?          uint32
|   |   |   +---rw clusterlist-value?  string

```

```

| |   +--rw bgp-originator-id?          uint32
+--ro bgp-route-create?                  enumeration
+--ro bgp-rt-state-info
| |   +--ro rib-current-state?          rib-state-def
| |   +--ro rib-last-state?            rib-state-def
| |   +--ro rib-rejected-reason?       enumeration
| |   +--ro not-preferred-reason?     enumeration
+--rw (bgp-evpn-route-type)?
| |   +--:(EVPN_AD_ROUTE)
| |   |   +--rw evpn-ad-route*
| |   |   [route-type route-distinguisher esi eth-tag-id]
| |   |   |   +--rw route-type          evpn-route-type
| |   |   |   +--rw route-distinguisher route-distinguisher-def
| |   |   |   +--rw esi                  string
| |   |   |   +--rw eth-tag-id          uint32
| |   |   +--:(EVPN_MAC_ROUTE)
| |   |   |   +--rw evpn-mac-route*
| |   |   |   [route-type route-distinguisher esi eth-tag-id
| |   |   |   mac-length mac-address ip-address-length ip-address]
| |   |   |   |   +--rw route-type          evpn-route-type
| |   |   |   |   +--rw route-distinguisher route-distinguisher-def
| |   |   |   |   +--rw esi                  string
| |   |   |   |   +--rw eth-tag-id          uint32
| |   |   |   |   +--rw mac-length          uint8
| |   |   |   |   +--rw mac-address         yang:mac-address
| |   |   |   |   +--rw ip-address-length   uint8
| |   |   |   |   +--rw ip-address         inet:ip-address
| |   |   |   |   +--rw label1?            uint32
| |   |   |   |   +--rw label2?            uint32
| |   |   +--:(EVPN_INC_MCAST_TREE)
| |   |   |   +--rw evpn-inc-mcast-tree-route*
| |   |   |   [route-type route-distinguisher eth-tag-id
| |   |   |   ip-address-length origin-router-ip-addr]
| |   |   |   |   +--rw route-type          evpn-route-type
| |   |   |   |   +--rw route-distinguisher route-distinguisher-def
| |   |   |   |   +--rw eth-tag-id          uint32
| |   |   |   |   +--rw ip-address-length   uint8
| |   |   |   |   +--rw origin-router-ip-addr inet:ip-prefix
| |   |   +--:(EVPN_ETH_SEGMENT_ROUTE)
| |   |   |   +--rw evpn-eth-segment-route*
| |   |   |   [route-type route-distinguisher esi
| |   |   |   ip-address-length origin-router-ip-addr]
| |   |   |   |   +--rw route-type          evpn-route-type
| |   |   |   |   +--rw route-distinguisher route-distinguisher-def
| |   |   |   |   +--rw esi                  string
| |   |   |   |   +--rw ip-address-length   uint8
| |   |   |   |   +--rw origin-router-ip-addr inet:ip-prefix
+--rw bgp-rib-out

```

```

+--rw bgp-route-type?          enumeration
+--rw route-admin-distance    uint16
+--rw bgp-attribute-list
|
|   +--rw bgp-origin?          enumeration
|   +--rw bgp-aspath
|   |   +--rw usascount?        uint16
|   |   +--rw ulrefcount?      uint16
|   |   +--rw asstring?        string
|   |   +--ro usascountfrcomp?  uint16
|   |   +--rw usastotalcount?  uint16
|   |   +--rw usas4pathlen?    uint16
|   |   +--rw as4pathvalue?    string
|   +--rw bgp-nexthop?        inet:ip-address
|   +--rw bgp-med?            uint32
|   +--rw bgp-localpref?      uint32
|   +--rw bgp-atomic-aggregate? uint32
|   +--rw bgp-agggregator
|   |   +--rw ulipaddress?      uint32
|   |   +--rw ulasnumber?      uint32
|   +--rw bgp-commattr
|   |   +--rw uscommsize?        uint16
|   |   +--rw ulrefcomm?        uint32
|   |   +--rw commattr-value?   string
|   +--rw bgp-extcommattr
|   |   +--rw custom-community
|   |   |   +--rw valid          boolean
|   |   |   +--rw insertion-point uint32
|   |   |   +--rw community-id   uint8
|   |   |   +--rw cost-id        uint32
|   |   +--rw usextcommsize?    uint16
|   |   +--rw ulrefcount?      uint32
|   |   +--rw extcommattr-value? string
|   +--rw bgp-clusterlist
|   |   +--rw uscluslen?        uint16
|   |   +--rw ulrefclus?        uint32
|   |   +--rw clusterlist-value? string
|   +--rw bgp-originator-id?  uint32
+--ro bgp-route-create?        enumeration
+--ro bgp-rt-state-info
|   +--ro rib-current-state?    rib-state-def
|   +--ro rib-last-state?      rib-state-def
|   +--ro rib-rejected-reason?  enumeration
|   +--ro not-preferred-reason? enumeration
+--rw (bgp-evpn-route-type)?
+--:(EVPN_AD_ROUTE)
|   +--rw evpn-ad-route*
|   |   [route-type route-distinguisher esi eth-tag-id]
|   |   +--rw route-type          evpn-route-type

```

```

|         +--rw route-distinguisher route-distinguisher-def
|         +--rw esi                      string
|         +--rw eth-tag-id                uint32
+---:(EVPN_MAC_ROUTE)
|   +--rw evpn-mac-route*
|     [route-type route-distinguisher esi eth-tag-id
|       mac-length mac-address ip-address-length ip-address]
|     +--rw route-type                    evpn-route-type
|     +--rw route-distinguisher route-distinguisher-def
|     +--rw esi                          string
|     +--rw eth-tag-id                    uint32
|     +--rw mac-length                    uint8
|     +--rw mac-address                    yang:mac-address
|     +--rw ip-address-length              uint8
|     +--rw ip-address                     inet:ip-address
|     +--rw label1?                       uint32
|     +--rw label2?                       uint32
+---:(EVPN_INC_MCAST_TREE)
|   +--rw evpn-inc-mcast-tree-route*
|     [route-type route-distinguisher eth-tag-id
|       ip-address-length origin-router-ip-addr]
|     +--rw route-type                    evpn-route-type
|     +--rw route-distinguisher route-distinguisher-def
|     +--rw eth-tag-id                    uint32
|     +--rw ip-address-length              uint8
|     +--rw origin-router-ip-addr          inet:ip-prefix
+---:(EVPN_ETH_SEGMENT_ROUTE)
|   +--rw evpn-eth-segment-route*
|     [route-type route-distinguisher esi
|       ip-address-length origin-router-ip-addr]
|     +--rw route-type                    evpn-route-type
|     +--rw route-distinguisher route-distinguisher-def
|     +--rw esi                          string
|     +--rw ip-address-length              uint8
|     +--rw origin-router-ip-addr          inet:ip-prefix

```

## 5. IANA Considerations

This draft includes no request to IANA.

## 6. Security Considerations

TBD.

## 7. Informative References

- [I-D.hares-i2rs-bnp-info-model]  
Hares, S. and Q. Wu, "An Information Model for Basic Network Policy", draft-hares-i2rs-bnp-info-model-00 (work in progress), September 2014.
- [I-D.hares-i2rs-info-model-policy]  
Hares, S. and W. Wu, "An Information Model for Basic Network Policy", draft-hares-i2rs-info-model-policy-03 (work in progress), July 2014.
- [I-D.ietf-i2rs-architecture]  
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-05 (work in progress), July 2014.
- [I-D.ietf-i2rs-rib-info-model]  
Bahadur, N., Folkes, R., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-03 (work in progress), May 2014.
- [I-D.ietf-netconf-restconf]  
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-02 (work in progress), October 2014.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3060] Moore, B., Ellessen, E., Strassner, J., and A. Westerinen, "Policy Core Information Model -- Version 1 Specification", RFC 3060, February 2001.
- [RFC3460] Moore, B., "Policy Core Information Model (PCIM) Extensions", RFC 3460, January 2003.
- [RFC3644] Snir, Y., Ramberg, Y., Strassner, J., Cohen, R., and B. Moore, "Policy Quality of Service (QoS) Information Model", RFC 3644, November 2003.
- [RFC5394] Bryskin, I., Papadimitriou, D., Berger, L., and J. Ash, "Policy-Enabled Path Computation Framework", RFC 5394, December 2008.

- [RFC5511] Farrel, A., "Routing Backus-Naur Form (RBNF): A Syntax Used to Form Encoding Rules in Various Routing Protocol Specifications", RFC 5511, April 2009.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

Authors' Addresses

Susan Hares  
Huawei  
7453 Hickory Hill  
Saline, MI 48176  
USA  
  
Email: shares@ndzh.com

Lixing Wang  
Huawei  
Beijing  
PRC  
  
Email: wanglixing@huawei.com

Shunwan Zhuang  
Huawei  
Beijing  
PRC  
  
Email: Zhuangshunwan@huawei.com

I2RS working group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 3, 2015

S. Hares  
Q. Wu  
Huawei  
July 2, 2014

An Information Model for Basic Network Policy  
draft-hares-i2rs-info-model-policy-03

Abstract

This document contains three information Models: Basic Network Policy (BNP IM), Policy-Based Routing (PBR), I2RS Local-Config (I2RS-LC IM). The I2RS-LC IM provides both an I2RS store of Policies plus a store of Policy Templates. The BNP IM has the following levels of Policy Hierarchy: Policy Set, Policy Group, Policy Rule, and conditional actions within the policy rule (conditional match and Action). The PBR IM, I2RS LC IM, BGP Information Model (BGP IM), Service Topology Information Model (SF-Topo IM), and the Service Forwarding Chaining IM (SFC IM) utilize and extend the BNP IM. This draft lists the extensions to the BNP IM that support these information models (PBR IM, I2RS LC IM, BGP IM, SSF-Topo IM and SFC-Policy IM).

The BNP IM is based on the concept of an extensible information model for representing policies. This concept is also found in the Policy Core Information Model (PCIM) (RFC3060) and the Quality of Service (QoS) Policy Information Model (QPIM) (RFC3644) and policy based routing.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 3, 2015.

## Copyright Notice

Copyright (c) 2014 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. Definitions and Acronyms . . . . .	5
3. Basic Network Policy Information Model (BNP IM) . . . . .	5
3.1. BNP IM Overview . . . . .	5
3.2. The Policy Set . . . . .	7
3.2.1. Policy Set Overview . . . . .	7
3.2.2. Policy-Set RBNF . . . . .	8
3.3. The Policy Group . . . . .	8
3.3.1. Policy Group Overview . . . . .	8
3.3.2. Policy-Group RBNF . . . . .	10
3.4. The Policy Rule . . . . .	11
3.4.1. Policy-Rule Overview . . . . .	11
3.4.2. Policy-Rule RBNF . . . . .	12
3.5. BNP IM Grammar . . . . .	16
4. The Policy Based Routing Information Model . . . . .	17
4.1. Policy Based Routing Overview . . . . .	17
4.2. PBR-RIB definition . . . . .	17
4.3. PBR Rule Component . . . . .	18
4.4. PBR QOS RBNF . . . . .	19
4.5. Relationship between PBR Rule Model and RIB Information Model . . . . .	21
4.6. PBR RBNF . . . . .	21
4.7. Remaining PRB Issues . . . . .	22
5. The I2RS Local Policy Information Model . . . . .	23
5.1. I2RS Local Policy IM Overview . . . . .	23
5.2. I2RS Local Policy IM RBNF . . . . .	23
6. Extensions to the Policy IM . . . . .	25
6.1. Extension to the RIB IM . . . . .	25
6.2. Extension from the BGP IM . . . . .	25
6.3. Extension from SFC Topology IM . . . . .	26
6.4. Extension from the SFC Traffic Filters . . . . .	26



7. IANA Considerations . . . . .	26
8. Security Considerations . . . . .	26
9. Informative References . . . . .	26
Authors' Addresses . . . . .	27

## 1. Introduction

The Interface to the Routing System (I2RS) provides read and write access to the information and state within the routing process within routing elements. The I2RS client interacts with one or more I2RS agents to collect information from network routing systems.

Processing of collected information at the I2RS agent may require the I2RS Agent to filter certain information or group pieces of information in order to reduce the data flow through the network to the I2RS client. Some applications that utilize the services of I2RS client may also wish to require specific data in response to network events or conditions based on pre-established rules. This functionality is necessary to meet the requirements of i2rs enabled services which include service-layer routing improvements, and control of traffic flows and exit points.

This document introduces a Basic Network Policy information model (BNP IM) to handle policies related to the network. This basic policy model can be easily extended beyond the basic functions. The [I-D.ietf-i2rs-architecture] suggests that associated with the i2RS RIB model there will be "Policy-based Routing (ACLs)" and RIB "policy controls". These basic policy functions can operate as part of this functional blocks providing the basic model for policy operators. This model can also be considered as the substance of the policy templates.

The BNP IM is extensible allowing other extensions to make the BNP IM policy adaptable to specific I2RS protocol features. This policy model can be linked with other information models such as the following:

- o Policy Base Routing Information model (PBR-IM) (Model in section 4),
- o I2RS RIB Informational Model (RIB IM) (see section 6) ([I-D.ietf-i2rs-rib-info-model])
- o BGP Informational Model (BGP IM) (see section 6) ([I-D.hares-i2rs-bgp-im])
- o Service Topology (see section 6) ([I-D.hares-i2rs-info-model-service-topo])

- o Service Forwarding Chaining Filters Information Mode (SFC IM) (see section 6) (ietf-hares-dunbar-i2rs-sfc-policy-im-00.txt)

The BNP IM model is a product of the industry approach to I2RS that standardizes on a few basic functions network functions to obtain quick deployment of initial I2RS RIB modules, and build on this success to create network functions. Additional I2RS modules add I2RS interfaces to policy-based routing, BGP, Service topology creation, Service Chaining functions, and policy templates.

This information model leverages previous work done on extensible information model for representing policies, for example, the Policy Core Information Model (PCIM) [RFC3060] [RFC3060], and an extension to this model to address the need for QoS management, called the Quality of Service (QoS) Policy Information Model (QPIM) [RFC3644] [RFC3644].

Most policy within routing and forwarding systems has become hierarchical with individual specific policies being grouped as a set policy. The hierarchical policy rule definition enhances policy readability and reusability. Groups of network policies have labels to aid operational use. Named groups of policy are easily identified and reused as blocks.

The Basic Network Policy information model contains the following three components:

#### Policy Group

Policy is described by a set of policy rules that may be grouped into subsets. A Policy group is used to provide a hierarchical policy definition that provides the model context or scope for sub-rule actions. The model context includes identity, scope, role, precedence, priority and security model. In a policy group policy rules and policy groups can be nested within other policy rules.

#### Policy Set

is a set of Policy Groups identified by a Policy Set Name.

#### Policy Rule

Policy Rule is represented by semantics "If Condition then Action", therefore condition and action comprise Policy Rule model.

This draft contains the following Informational Models

#### Basic Network-Policy Information Model (BNP IM)

is generic network policy model. It can be thought of as a coherent set of rules to administer, manage, and control access to network resources and defines a network policy at its most general level of abstraction. It models aspects such as actions and conditions that constitute a policy element relationship, as well as operators contained in the both condition and action that can either be used to overwrite an old value of the variable or imply match relationship.

#### Policy Based Routing Information Model (PBR IM)

defines information that allows the network administrator to forward the packet based on other criteria than the destination address in the packet.

#### I2RS Local Config Information Model (I2RS-LC IM)

defines I2RS Local Configuration database kept in the I2RS Agent that can be leveraged to quickly set-up policies via the I2RS Agent. This local configuration store contains basic network policies and network templates, and provides quick local access to policies rather than transfer the policies down from the I2RS Client prior to enacting a policy via I2RS interface.

## 2. Definitions and Acronyms

IGP: Interior Gateway Protocol

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

CLI: Command Line Interface

SNMP: The Simple Network Management Protocol

NETCONF: The Network Configuration Protocol

RBNF: Routing Backus-Naur Form

## 3. Basic Network Policy Information Model (BNP IM)

### 3.1. BNP IM Overview

I2RS needs its own implicit and explicit policy. This section provides an overview of the network policy model. The network policy

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

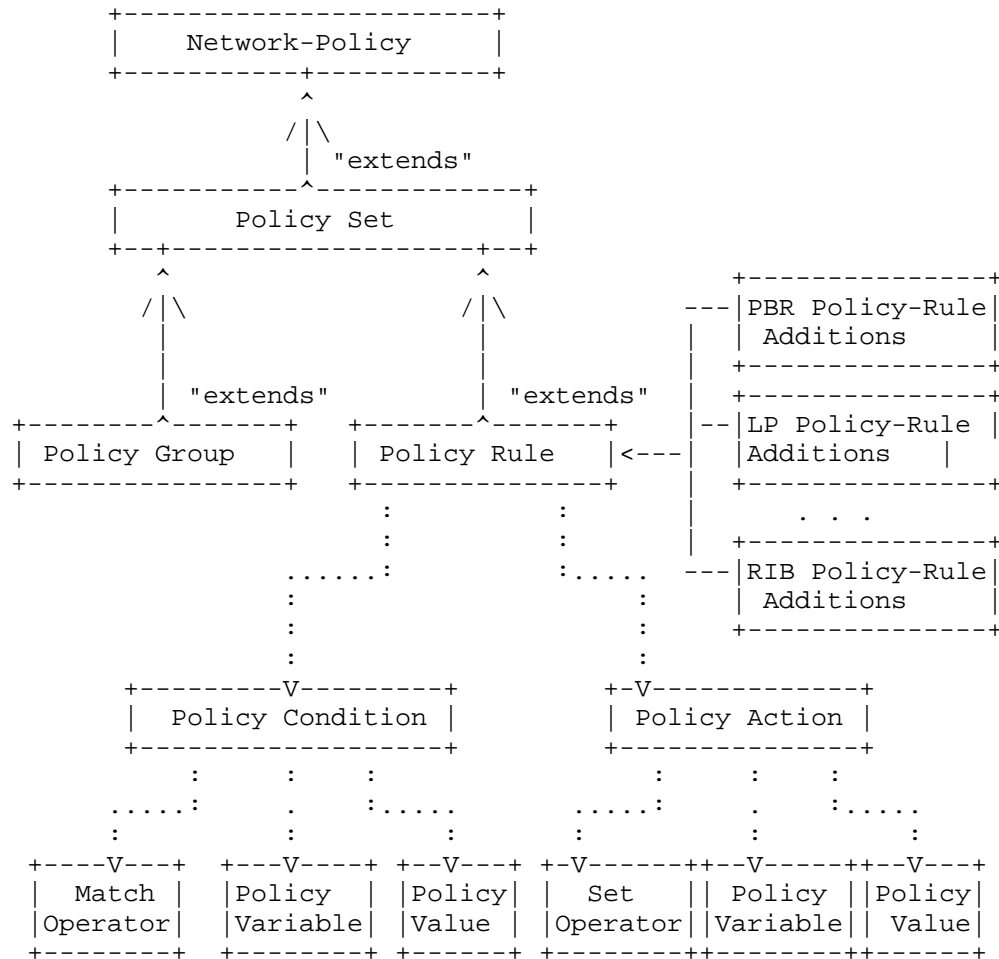


Figure 1: Overall model BNP IM structure

`Network_policy` - contains sets of policies.

`Policy-Set` - is introduced to provide an abstraction for a set of rules. it is inserted into the inheritance hierarchy above both `Policy-Group` and `Policy-Rule`.

`Policy-Group` - defines the basic network policy Group model which combines the a list of `Policy-Rules`.

Policy Rule is represented by semantics "If Condition then Action", therefore condition and action comprise Policy Rule model.

- o Condition models the elementary match operation "<variable> match <value>".
- o Action models the elementary set operation. "SET <variable> TO <value>".

In Condition model, the 'Match' operator is usually implied while in the action model, the 'Set operator is explicitly used.

Policy-Sets, Policy-Groups, and Policy-Rules have basic functionality (Policy-Basic IM) plus extensions defined by specific Information Models such as:

- the PBR Information Model (PBR IM) (contained in this document),
- the I2RS\_Local\_Policy Model (LP IM) (contained in this document),
- the RIB Information Model (RIB IM)  
([I-D.ietf-i2rs-rib-info-model]),
- the BGP Information Model (BGP-IM) ([I-D.hares-i2rs-bgp-im]),
- the Traffic Steering Information Model  
([I-D.hares-i2rs-info-model-service-topo]),
- the SFC Information Model (SFC IM) (ietf-hares-dunbar-i2rs-sfc-policy-im-00.txt)
- the MPLS LDP Information Model (MPLS LDP IM) as templates for policy.

I2RS Client-Agents Information Models MAY support only the Policy-Basic IM, or MAY support any additional specific information models.

Each level of the Policy hierarchy (Policy-Set, Policy-Group, and Policy-Rules have both a read and write scope

### 3.2. The Policy Set

#### 3.2.1. Policy Set Overview

The PolicySet structure has the following elements:

- o Policy-Set\_Name - Unique Name for Policy Set

- o Policy-Group is introduced to provide an abstraction for a set of rules. It is derived from Policy, and it is inserted into the inheritance hierarchy above both PolicyGroup and PolicyRule. This reflects the additional structural flexibility and semantic capability of both subclasses.

### 3.2.2. Policy-Set RBNF

Figure 2 - Policy Set RBNF

```
<Network_policy> ::= (<Policy_Set> ...)
<Policy-Set> ::= <Policy-Set-Name>

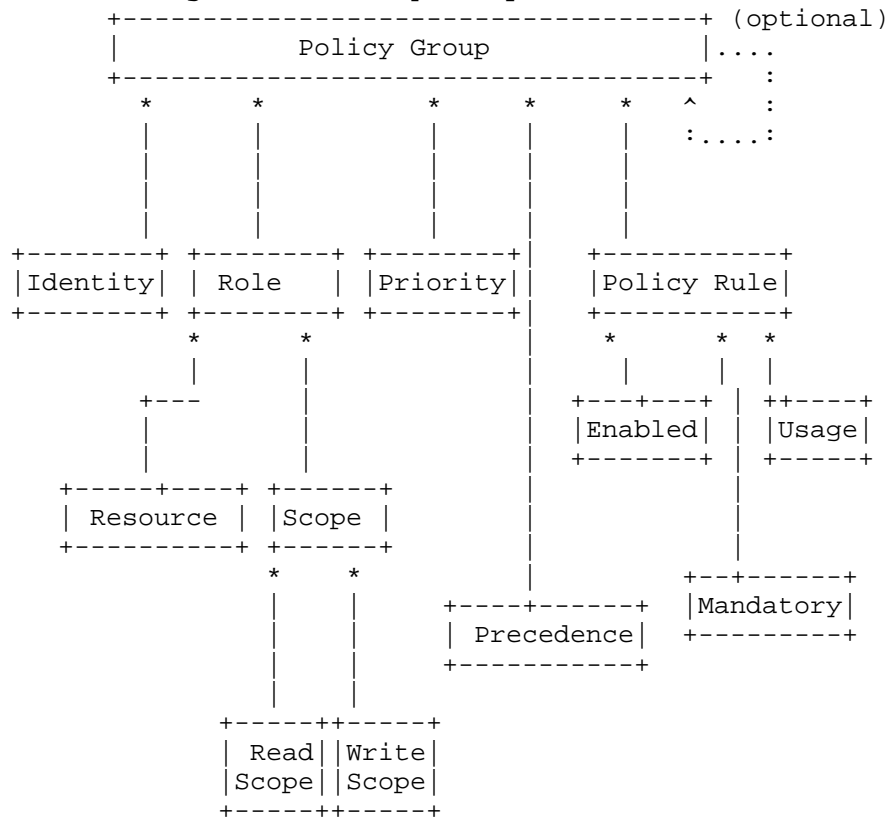
<Policy-Group_list> ::=
    (<Policy-Group> ...)
    (<Policy-Rule> ...)
    (<local_Config>)
    (<PBR_rule>)
```

## 3.3. The Policy Group

### 3.3.1. Policy Group Overview

In order to provide hierarchical policy definition and associate policy rule with other constraint, the basic policy group model needs to be defined. The corresponding extensions are introduced in a component, whose structure is informally depicted in the following diagram.

Figure 3 - Policy Group



The basic information model works as follows: Within the policy group information model, hierarchy is used to model context or scope for the sub-rule actions. A policy group contains Identity, scope, priority, precedence, and policy rule. Optionally, the policy group can contain a list of policy groups.

The elements of the Policy Group information model are as follows:

- o Each policy group is captured in its own list, distinguished via a identity, role, priority, precedence.
- o A policy group has a certain role, such as resource or scope. A policy group can even have multiple roles simultaneously. The role, are captured in the list of "role" component.
- o A policy role has a certain Scope, such as read scope or write scope. A policy group can even have multiple scope

simultaneously. The scope, or scopes, are captured in the list of "scope" components.

- o A policy has a certain priority, such as priority 0-255. A policy can only have one priority. The priority is captured in the list of "priority" component.
- o A policy rule can inherit properties (e.g., identity, role, priority, precedence) from policy group. A policy rule also can have its own properties, e.g., enabled, mandatory, usage.
- o The policy, policy group elements can be extended with policy-specific components (policy-extensions, policy-group-extension respectively).

### 3.3.2. Policy-Group RBNF

A more formal depiction in RBNF format follows below

Figure 4 - Policy-Group RBNF

```

<Policy-Group> ::= <Policy-Group_Identity>
                  <Policy-Group_Roles>
                  <Policy-Group_priority>
                  <Policy-Group_precedence>
                  (<Policy-Rule-list>)
                  [<Supporting-Policy-Group>]
                  [<Policy-Group-Extension>]

<Policy-Group_Identity> ::= <Policy-Group-Name>
                           [<Policy-Group-Secure-Identity>]
<Policy-Group-priority> ::= INTEGER (0..255);
<Policy-Group-precedence> ::= INTEGER (0..250);
<Policy-Rule-list> ::= ((<Policy-Rule> <Policy-Rule-Status>) ...)

<Policy-Rule-Status> ::= <POLICY-RULE-ENABLE>
                       [<POLICY-RULE-MANDATORY>]
                       [<Policy-Rule_usage>]

<Policy-Rule_usage> ::= <Policy-Rule-REFCNT>

<Policy-Group-Roles> ::= (<Policy-Group-Role> ...)
<Policy-Group-Role> ::= <Node-RESOURCES> | <Policy-Group-Scope>
<Node-RESOURCES> ::= [<I2RS_AGENT_RESOURCE>]

<Policy-Group-Scope> ::= (<READ_SCOPE> <Policy-Group_Read_Scope>)
                        | (<WRITE_SCOPE> <Policy-Group_Write_Scope>)

```



```

<Policy-Group_Read_Scope> ::= <Policy-Group_Read_Scope_Type>
                               [<RIB-IM_READ_list>]
                               [<BGP-IM-READ_list>]

<Policy-Group_Read_Scope_Type> ::= <RIB-IM_READ_SCOPE_TYPE>
                                   | <BGP-IM_READ_SCOPE_TYPE>

<Policy-Group_Write_Scope> ::= <Policy-Group_Write_Scope_Type>
                               [<RIB-IM_WRITE_list>]
                               [<BGP-IM-WRITE_list>]

<Policy-Group_Write_Scope_Type> ::= <RIB-IM_WRITE_SCOPE_TYPE>
                                   | <BGP-IM_WRITE_SCOPE_TYPE>

<Supporting-Policy-Group> ::= <SUPPORT-POLICY-GROUP> (
                               <Policy-Group> ...)

<Policy-Group-Extension> ::= ... /* Vendor Specific Policy */

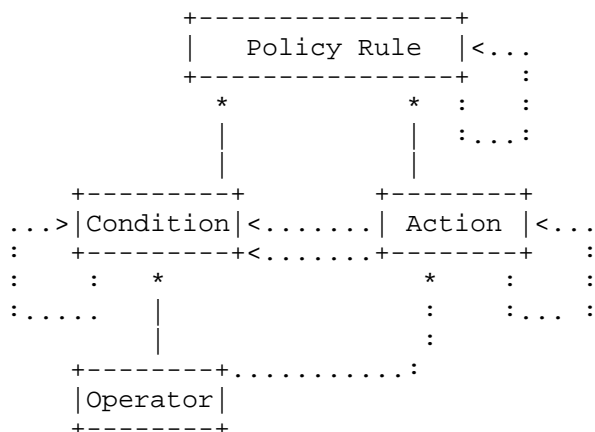
```

### 3.4. The Policy Rule

#### 3.4.1. Policy-Rule Overview

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

Figure 5 - Policy Rule



Roughly speaking, the basic information model works as follows: A policy rule contains conditions and actions. Each condition or each action in turn contains operator. A operator connects variable and value in the action or condition. Condition can map onto and be supported by other condition, while action can map onto and be supported by other actions. Policy rule can map onto other, policy rules.

The elements of the Policy Rule information model are as follows:

- o A policy can in turn be part of a hierarchy of policies, building on top of other policies. Each policy is captured in its own level, distinguished via a policy-identity.
- o Policy rule inherit scope from policy group. A policy rule has a certain Scope, such as read scope or write scope. A policy rule can even have multiple scope simultaneously. The scope, or scopes, are captured in the list of "scope" components.
- o Furthermore, a policy rule contains conditions and actions, each captured in their own list.
- o A condition contains a variable and a value and use a match operator, to connect variable with value. An examples of an operator might be a " IP ADDRESS AS RESOLVED BYDNS" or "Set to a member". Also, a condition can in turn map onto other condition in an underlay policy. This is captured in list "supporting-condition".
- o An action contains a variable and a value. An action uses Set operator to connect variable with value. Analogous to a node, an action can in turn map onto other actions in an underlay policy. This is captured in list "supporting-action".
- o The policy, condition, action and operator elements can be extended with policy-specific components (policy-extensions, condition-extension, action-extension and operator-extension respectively).

#### 3.4.2. Policy-Rule RBNF

The information model for the Network-policy component is more formally shown in RBNF below:

Figure 6 Policy Rule RBNF

```
<Policy-Rule> ::= <Policy-Rule_identity>  
                  <Policy-Rule_priority>
```

```

        <Policy-Rule_precedence>
        <Policy-Rule_Roles>
        (<Policy-Rule_Condition>
        <Policy-Rule_Action> ...)
        <Policy-Rule_Security_model>
        [<Policy-Rule_rule_extensions>]

<Policy-Rule_identity> ::= <Policy-Rule-Name>
                           [<Policy-Rule-Secure-Identity>]
<Policy-Rule_priority> ::= INTEGER (0..250);
<Policy-Rule_precedence> ::= INTEGER (0..250);

<Policy-Rule_Roles> ::= (<Policy_Rule_Role> ...);
<Policy-Rule_Role> ::= <RESOURCES> |
                      <Policy_Rule_Scope>

<RESOURCES> ::= [<I2RS_AGENT_RESOURCE>]
<Policy-Rule_Scope> ::= (<READ_SCOPE>
                        <Policy-Rule_Read_scope>)
                        | (<WRITE_SCOPE>
                        <Policy-Rule_Write_scope>)

<Policy-Rule_Read_scope> ::= ((<BNP_READ_SCOPE_TYPE>
                             <BNP_READ_SCOPE_list>) ...)
                        | [<Policy-Rule_Read_Scope_External>]

<Policy-Rule_Write_scope> ::= ((<BNP_WRITE_SCOPE_TYPE>
                             <BNP_WRITE_SCOPE_list>)...)
                        [<Policy-Rule_Write_Scope_External>]

<Policy-Rule_Condition> ::= <Policy-Rule_Match_node>
                           (<Policy-Rule_Match_value> ...)
                           [<Policy-Rule_mode>]
                           [<Policy-Rule_Match_Operator>]
                           [<Policy-Rule_Condition_extension>]

<Policy-Rule_Match_node> ::= [<Policy-Rule_Match_Node_BNP-IM>]
                           | [<Policy-Rule_Match_node_external>]

<Policy-Rule_Match_value> ::= [<Policy-Rule_Match_Value_BNP-IM>]
                           | [<Policy-Rule_Match_Value_external>]

<Policy-Rule_mode> ::= PERMIT | DENY ;
<Policy-Rule_Match_operator_external> ::=
                           [<Policy-Rule_Match_Operator_BNP-IM>]
                           | [<Policy-Rule_Match_Operator_external>]

```

```

<Policy-Rule-action> ::= <Policy-Rule_Action_variable>
                        <Policy-Rule_Action_value>
                        <Policy-Rule_Set-Operator>
                        [ <Policy-Rule_action-extension> ]

<Policy-Rule_Security-Model> ::= <First-Matching>
                                | <All-Matching>]

<Policy-Rule_rule_extension> ::=
                                <I2RS-LC-policy_rule_extensions>

<Policy-Rule-action> ::= <Policy-Rule_Action_variable>
                        <Policy-Rule_Action_value>
                        <Policy-Rule_Set-Operator>
                        [ <Policy-Rule_action-extension> ]

<Policy-Rule_Action_variable> ::= <Policy-Rule_Action_var>
                                (<Policy-Rule_Action_value> ...)
                                [ <Policy-Rule_Set_Operator> ]
                                [ <Policy-Rule_Action_extension> ]

<Policy-Rule_Action_var> ::= [ <Policy-Rule_Action_Vars_BNP-IM> ]
                             | [ <Policy-Rule_Action_external> ]

<Policy-Rule_Action_value> ::= [ <Policy-Rule_Action_Vars_BNP-IM> ]
                              | [ <Policy-Rule_Action_external> ]

<Policy-Rule_Set_Operator> ::= [ <Policy-Rule_Set_Operator_BNP-IM> ]
                              | [ <Policy-Rule_Set_Operator_external> ]

<Policy-Rule-action-extension> ::=
                                [ <Policy-Rule_act_ext_BNP-IM> ]
                                | [ <Policy-Rule_act_ext_external> ]

<Policy-Rule-Match-Operator-Policy-IM> ::= <IS-SET-MEMBER'>
                                           | <IN-INTEGGER-RANGE>
                                           | <IP-ADDRESS-AS-RESOLVED-BY-DNS>
                                           | <Policy_IM-Match-Operator-extension>

<Policy-Rule_condition_extension> ::=
                                <Policy_Rule_condition_ext-BNP-IM>
                                [ <Policy-Rule_Condition_ext_external> ]

/* these scopes besides RIB IM are defined in each IM */

<PR_Read_Scope_RIB_IM> ::= <RIB-IM_READ_SCOPE_TYPE>
                           <RIB-IM_READ_list>

```

```

<PR_Read_Scope_RIB_IM> ::= <RIB-IM_READ_SCOPE_TYPE>
                                <RIB-IM_READ_list>

<RIB-IM_READ_list> ::= [ <RIB-IM-Tree-Match> ... ]
<RIB-IM_WRITE_list> ::= [ <RIB-IM-Tree-Match> ... ]
<RIB-IM-Tree-Match> ::= <RIB-IM-Match-routing-instance>
                                <RIB-IM-Match-interface-list>
                                <RIB-IM-Match-rib_list>
                                <RIB-IM-match-route-list>

/* extensions to other IM */

/* External Read and Write Scope */
<Policy-Rule_Read_Scope_External> ::=
    [ <PR_Read_Scope_RIB_IM> ]
    [ <PR_Read_Scope_BGP_IM> ]
    [ <PR_Read_Scope_PBR_IM> ]
    [ <PR_Read_Scope_I2RSLC_IM> ]
    [ <PR_Read_Scope_STopo_IM> ]
    [ <PR_Read_Scope_SFC-Policy_IM> ]

<Policy-Rule_Write_Scope_External> ::=
    [ <PR_Write_Scope_RIB_IM> ]
    [ <PR_Write_Scope_BGP_IM> ]
    [ <PR_WriteScope_PBR_IM> ]
    [ <PR_Read_Scope_I2RSLC_IM> ]
    [ <PR_Read_Scope_STopo_IM> ]
    [ <PR_Read_Scope_SFC-PolicyIM> ]

/* External Rule Conditionals */
<Policy-Rule_Match_node_external> ::=
    [ <Policy-Rule_Match_Node_RIB-IM> ]
    | [ <Policy-Rule_Match_Node_PBR-IM> ]
    | [ <Policy-Rule_match_Node_I2RSLC-IM> ]
    | [ <Policy-Rule_Match_Node_BGP-IM> ]
    | [ <Policy-Rule_Match_Node_STopo-IM> ]
    | [ <Policy-Rule_Match_Node_SFC-Policy-IM> ]

<Policy-Rule_Match_Value_external> ::=
    [ <Policy-Rule_Match_Value_RIB-IM> ]
    | [ <Policy-Rule_Match_Value_PBR-IM> ]
    | [ <Policy-Rule_Match_Value_I2RSLC-IM> ]
    | [ <Policy-Rule_Match_Value_BGP-IM> ]
    | [ <Policy-Rule_Match_Value_STopo-IM> ]
    | [ <Policy-Rule_Match_Value_SFC-Policy-IM> ]

<Policy-Rule_Match_operator_external> ::=
    [ <Policy-Rule_Match_Operator_RIB-IM> ]

```

```

| [<Policy-Rule_Match_Operator_PBR-IM>]
| [<Policy-Rule_Match_Operator_I2RSLC-IM>]
| [<Policy-Rule_Match_Operator_BGP-IM>]
| [<Policy-Rule_Match_Operator_STopo-IM>]
| [<Policy-Rule_Match_Operator_SFC-Policy-IM>]

<Policy-Rule_Action_value_external> ::=
| [<Policy-Rule_Action_Values_RIB-IM>]
| [<Policy-Rule_Action_Values_PBR-IM>]
| [<Policy-Rule_Match_Operator_I2RSLC-IM>]
| [<Policy-Rule_Action_Values_BGP-IM>]
| [<Policy-Rule_Set_Operator_STopo-IM>]
| [<Policy-Rule_Set_Operator_SFC-Policy-IM>]

<Policy-Rule_Set_Operator_external> ::=
| [<Policy-Rule_Set_Operator_RIB-IM>]
| [<Policy-Rule_Set_Operator_PBR-IM>]
| [<Policy-Rule_Match_Operator_I2RSLC-IM>]
| [<Policy-Rule_Set_Operator_RIB-IM>]
| [<Policy-Rule_Set_Operator_BGP-IM>]
| [<Policy-Rule_Set_Operator_STopo-IM>]
| [<Policy-Rule_Set_Operator_SFC-Policy-IM>]

<Policy-Rule_act_ext_external> ::=
| [<Policy-Rule_extension_RIB-IM>]
| [<Policy-Rule_act_ext_PBR-IM>]
| [<Policy-Rule_act_ext_I2RSLC-IM>]
| [<Policy-Rule_act_ext_RIB-IM>]
| [<Policy-Rule_act_ext_BGP-IM>]
| [<Policy-Rule_act_ext_STopo-IM>]
| [<Policy-Rule_act_ext_SFC-Policy-IM>]
| [<I2RS_Vendor-Rule_act_ext>]/* other I2RS IM */

```

### 3.5. BNP IM Grammar

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

```

<basic-network_policy_in> ::= (<policy-set> ...)
<basic-network_policy_out> ::= (<policy-set> ...)
<network-policy_rules_list> ::= (<policy-rule>...)

```

## 4. The Policy Based Routing Information Model

### 4.1. Policy Based Routing Overview

Policy based Routing is a technique used to make routing decisions based on policies set by the network administrator. PBR enables network administrator to forward the packet based on other criteria than the destination address in the packet, which is used to lookup an entry in the routing table.

The policy based routing problem can be viewed as a resource allocation problem that incorporates business decision with routing. Policy based routing provides many benefits, including cost saving, load balancing and basic QoS.

Routing decisions in policy based routing are based on several criteria beyond destination address, such as packet size, application, protocol used, and identity of the end system. Policy constraints are applied before applying QoS constraints since policy constraint overrides QoS constraint. Policy constraints may be exchanged by routing protocols while updating routing information.

The I2RS use cases which benefit from PBR are:  
[I-D.white-i2rs-use-case] and  
[I-D.krishnan-i2rs-large-flow-use-case],

PBR-Rules extends from Policy Basic Rule with a set of condition, action and attributes. Routing decisions in policy based routing are based on several criteria beyond destination address, such as packet size, application, protocol used, and identity of the end system.

### 4.2. PBR-RIB definition

One routing instance (named by an INSTANCE\_NAME) can contain multiple PBR RIBs, and is associated with a set of interfaces, and a ROUTER-ID. The entries associated with each routing instance relating to the PBR are:

- o INSTANCE NAME
- o interface\_list
- o PRB RIB list - with each entry having an order set of routes
- o PRB Default RIB - default forwarding FIB.
- o ROUTER-ID

Each PBR RIB has the following:

- o PRB RIB NAME
- o PBR Route-entry

The Route entry in a PRB has the following information:

- o match field - as in the RIB IM route
- o order\_list PBR route list with each entry having: a) next-hops, b) PBR route attributes, and c) vendor-attributes

The PRB route attributes include QOS Attributes as show in the policy list below.

#### 4.3. PBR Rule Component

A PBR rule is constructed using condition, action and attributes that are inherited from Policy Group Component.



Figure 7 - PBR Policy Rule

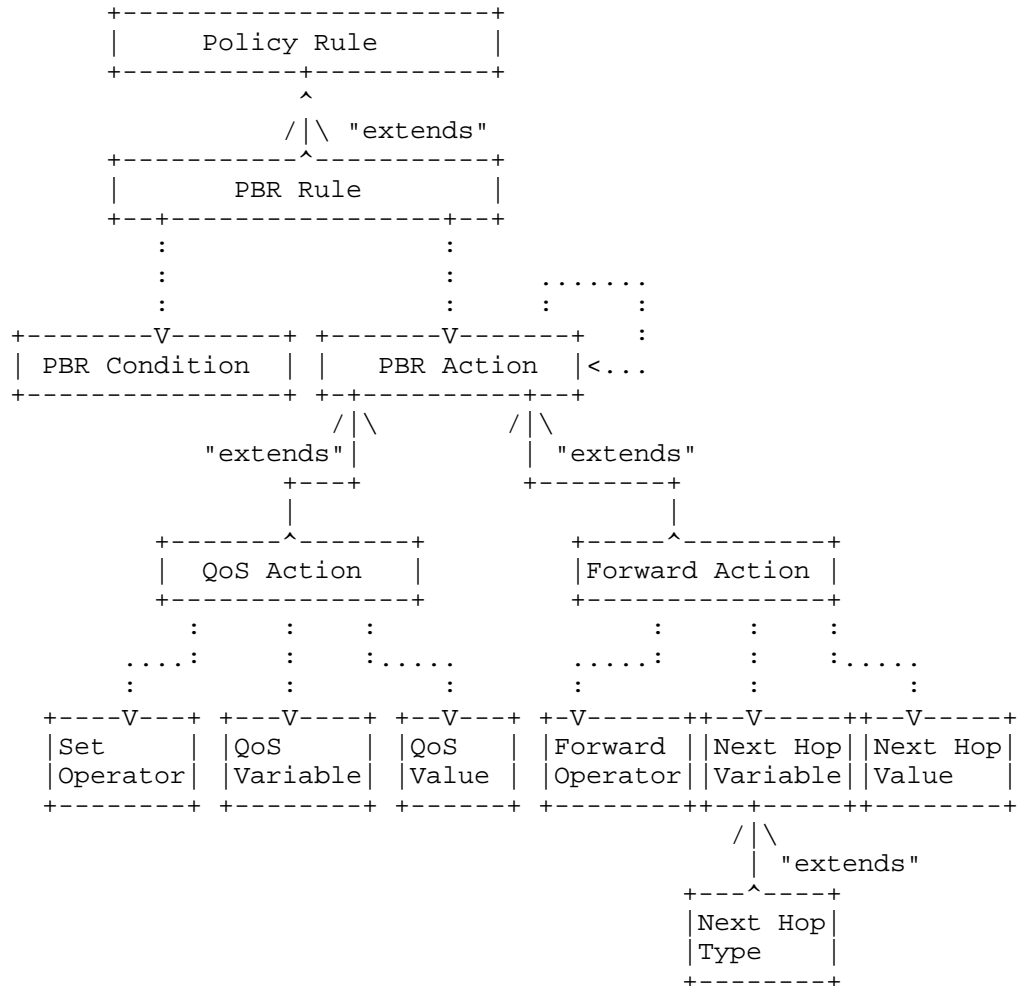


Figure 3: Policy based routing IM structure

#### 4.4. PBR QOS RBNF

The PBR QOS RBNF is below

Figure 8 - PRB QOS RBNF

```

/* policy rules */
<Policy-Rule_Match_Node_PBR-IM> ::= <IPv4_QoS_Node_Matches>
| <IPv6_QoS_Node_Matches>

```

```

<Policy-Rule_Match_Value_PBR-IM> ::= <IPv4_QoS_Value_Matches>
| <IPv6_QoS_Value_Matches>

<IPv4_QoS_Node_matches> = <IPv4-QoS_Matches>
<IPv6_QoS_Node_matches> = <IPv6-QoS_Matches>
<IPv4_QoS_Value_matches> = <IPv4-QoS_Matches>
<IPv6_QoS_Value_matches> = <IPv6-QoS_Matches>

<IPv4-QoS_Matches> ::= [<IPv4-SRC>]
| [<IPv4_DST>]
| [<IPv4_Proto>]
| [<IPv4_TOS-DSCP>]
| [<IPv4_ICMP_field>]
| [<IPv4_length>]

<IPv6-QoS_Matches> ::= [<IPv6-SRC>]
| [<IPv6_DST>]
| [<IPv6_Proto>]
| [<IPv6_Flow>]
| [<IPv6_length>]

<Policy-Rule_Match_Operator_PBR-IM> ::= [<Longest-prefix>]
| [<Exact>]
| [(<IPv4-RANGE> <IPv4-Low> <IPv4-High>)]
| [(<IPv6-RANGE> <IPv6-Low> <IPv6-High>)]
| [(<LENGTH-Range> <LENGTH_Low> <LENGTH_High>)]

<IPv4_low> ::= <IPv4-Prefix>
<IPv4_high> ::= <IPv4-Prefix>
<IPv6_low> ::= <IPv6-Prefix>
<IPv6_high> ::= <IPv6-Prefix>

<Policy-Rule_Action_value_PBR-IM> ::= [<QoS_action>]
| [<FWD_action>]

<QoS_action> ::= <QoS_IP-TOS_Set>
| <QoS_DSCP>

<FWD_action> ::= <Drop_packet>
| <Drop_Packet_ICMP>
| (<Forward_Specific> <next-hop>)
| (<Forward_Default>)

<Policy-Rule_Set_Operator_PBR-IM> ::= [<SET_QOS_BITS>]
| [<SET_FWD_ACTION>]

```

#### 4.5. Relationship between PBR Rule Model and RIB Information Model

As described in [I-D.ietf-i2rs-rib-info-model], each Routing instance contains a collection of RIBs, interfaces, and routing parameters including the following:

- o The set of interfaces indicates which interfaces are associated with this routing instance.
- o The RIBs specify how incoming traffic is to be forwarded based on destination.
- o the routing parameters control the information in the RIBs/PIBs.

PIB and RIB can not be used at the same time, which means:

- o If a router doesn't support policy based routing, a router MUST use rib and MUST not use PIB.
- o If a router supports policy based routing,
  - \* PIB is used if several criteria beyond destination address is matched.
  - \* RIB is used if several criteria beyond destination address is not matched.

Policy constraints information either comes from RSVP,BGP/IGP, or comes from manual configuration or policy configuration tool. Therefore PBR uses from the RIB IM:

- o Interface-list: The interface list contains a list of identifiers, with each identifier uniquely identifying an interface.
- o Origin: an indication used to identify from which protocols (e.g., ISIS, OSPF, BGP, I2RS, CLI etc.) the policy based route is.

#### 4.6. PBR RBNF

Figure 9 - PBR RBNF  
/\* pbr defintion \*/

```

<pbr-routing-instance> ::= <PBR_INSTANCE_NAME>
                        [<interface-list>] <pbr-rib_list>
                        [<prb-default-rib>]
                        [<Router-ID>]

<pbr-rib-list> ::= <PRB_RIB_NAME>
                  <PRB_rib_family>
                  (<prb_rib> ...)

<PRB_rib_family> ::= <IPv4_PRB_FAMILY>
                  | <IPv6_PRB_FAIMILY>

<prb_rib> ::= <PRB_RIB_NAME>
              <PRB_rib_family>
              <prb_policy_set>

<prb_policy_set< ::= <BNP_policy_set>
                    <RIB_policy_set_extensions>
                    <PRB_policy_set_extensions>
                    <BGP_policy_set_extensions>
                    <QOS_policy_set_extensions>

<prb_policy_set_exetensions> ::=
    <Policy-Rule_Match_Node_PBR-IM>
    <Policy-Rule_Match_Value_PBR-IM>
    <Policy-Rule_Match_Operator_PBR-IM>
    <Policy-Rule_Action_value_PBR-IM>
    <Policy-Rule_Set_Operator_PBR-IM>

```

#### 4.7. Remaining PRB Issues

Policy based routing MUST tackle the following difficult questions:

- o How is policy management strategy selected? Centralized or distributed.
- o At which point in a network domain are policy constraints checked and enforced? i.e., policy coverage, here policy constraint can be exchanged by routing protocol?
- o How are policy constraints exchanged within a domain?
- o How is policy data stored, refreshed and retrieved from policy repository?

- o How are policy rule conflicts avoided?

## 5. The I2RS Local Policy Information Model

### 5.1. I2RS Local Policy IM Overview

The Local Policy Information Model (LB IM) stores I2RS policy and policy templates that are used across many I2RS modules. The LB IM stores a set of policy and a set of policy templates. This section defines the LB IM extensions needed to the Basic Policy set at the Policy-set, Policy-Group, and Policy-Rule level. It also defines the optional extensions to this LP policy model as:

- o PBR IM,
- o RIB IM,
- o BGP IM,
- o Traffic Steering (TS IM)
- o SFC Filter (SFC-F IM),
- o Basic Route Templates IM

The key benefit of the I2RS Local Configure Information Model (Local- is that it provides a place to store I2RS policy, and I2RS templates. The LB IM MAY allow for: a) re-use of these policy templates across multiple I2RS client-I2RS agent sessions, b) storing of some policy into permanent configuration store

### 5.2. I2RS Local Policy IM RBNF

Figure 10 - Local I2RS Policy Story extensions

```

<i2rs-Local-Policy> ::= <I2RS_LOCAL_POLICY_STORE>
                        <Policy-Set>
                        <Policy-Templates>

<i2rs-condition-extension> ::= [<I2RS-Client-Agent_transport_list>]
                                [<I2RS_IPADDRESS_DNS_resolve>]
                                [<Policy-Template_match>]

<I2RS-Client-Agent_transport_list> ::= (< i2rs_transport > ...)
<i2rs_transport> ::= [TCP][SCTP][SSL];

/* these actions allow local I2RS configure store could
 * write to local Node policy store */

<i2rs-action-extension> ::= [<i2rs_write_policy_config_store>]
                            [<i2rs_write_bgp_config_store>]
                            ...

```

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

- o A local policy rule can in turn be part of a hierarchy of policies, building on top of other policies. Each local configuration policy is captured in its own level, distinguished via a policy identity.
- o A local policy rule inherit scope from policy group. A local policy rule has a certain Scope, such as read scope or write scope. A local policy rule can even have multiple scope simultaneously. The scope, or scopes, are captured in the list of "scope" components.
- o Furthermore, a local policy contains conditions and actions, each captured in their own list.
- o A condition contains a variable and a value and use a match operator, to connect variable with value. An examples of an operator might be a " IP ADDRESS AS RESOLVED BYDNS" or "Set to a member". Also, a condition can in turn map onto other condition in an underlay policy. This is captured in list in "supporting-condition".
- o An action contains a variable and a value. An action uses Set operator to connect variable with value. Analogous to a node, an action can in turn map onto other actions in an underlay policy. This is captured in list "supporting-action".

- o The local policy, condition, action and operator elements can be extended with policy-specific components (condition-extension, action-extension and operator-extension respectively).

## 6. Extensions to the Policy IM

### 6.1. Extension to the RIB IM

Figure 11 - RIB Information Model Extensions

```

<RIB-IM_READ_list> ::= [<RIB-IM-Tree-Match ...>
<RIB-IM_WRITE_list> ::= [<RIB-IM-Tree-Match ...>
<RIB-IM-Tree-Match> ::= <RIB-IM-Match-routing-instance>
                        <RIB-IM-Match-interface-list>
                        <RIB-IM-Match-rib_list>
                        <RIB-IM-match-route-list>

/* BGP Info Module Tree Match */
<BGP-IM_READ_list> ::= [<BGP-IM-Tree-Match ...>
<BGP-IM_WRITE_list> ::= [<BGP-IM-Tree-Match ...>

<BGP-IM-Tree-Match> ::= <BGP-IM-Tree-Match-protocol-instance>
<BGP-IM-Match-Protocol-instance> ::= (<BGP_protocol> ...)

<prb_rib> ::= <bgp_route_list>

<bgp_route_list> ::= (<bgp_route> ...)
<bgp_route> ::= <BGP_ROUTE_TYPE>
                <bgp_route_prefix>
                <bgp_attribute_list>
                <bgp_route_create>
                <bgp_rt_state_info>

<basic-network_policy_in> ::= (<policy-set> ...)
<basic-network_policy_out> ::= (<policy-set> ...)
<network-policy_rules_list> ::= (<policy-rule>...)

```

### 6.2. Extension from the BGP IM

Figure 12 - BGP Information Model Extensions

```

<BGP-IM_READ_list> ::= [<BGP-IM-Tree-Match ...>
<BGP-IM_WRITE_list> ::= [<BGP-IM-Tree-Match ...>
<BGP-IM-Tree-Match> ::= <BGP-IM-Tree-Match-protocol-instance>
<BGP-IM-Match-Protocol-instance> ::= (<BGP_protocol> ...)

```

### 6.3. Extension from SFC Topology IM

Figure 13 - SFC Topology Information Model Extensions

```
/* what part of the STopo Model can access */  
  
<STopo-IM_READ_list> ::= [<STopo-IM-Tree-Match ...]  
<STopo-IM_WRITE_list> ::= [<STopo-IM-Tree-Match ...]  
<STopo-IM-Tree-Match> ::= <STopo-IM-Tree-Match-protocol-instance>  
<STopo-IM-Match-Protocol-instance> ::= (<STopo_protocol> ...)
```

### 6.4. Extension from the SFC Traffic Filters

Figure 14 - Traffic Steering Information Model Extensions

```
/* what part of the STopo Model can access */  
  
<SFC-Policy-IM_READ_list> ::= [<SF-Policy-IM-Tree-Match ...]  
<SFC-Policy-IM_WRITE_list> ::= [<SF-Policy-IM-Tree-Match ...]  
<SFC-Policy-IM-Tree-Match> ::= <SF-Policy-IM-Tree-Match-protocol-instance>  
<SFC-Policy-IM-Match-Protocol-instance> ::= <SF_instance_list>
```

## 7. IANA Considerations

This draft includes no request to IANA.

## 8. Security Considerations

TBD.

## 9. Informative References

- [I-D.atlas-i2rs-policy-framework]  
Atlas, A., Hares, S., and J. Halpern, "A Policy Framework for the Interface to the Routing System", draft-atlas-i2rs-policy-framework-00 (work in progress), February 2013.
- [I-D.hares-i2rs-bgp-im]  
Hares, S., Wang, L., and S. Zhuang, "An I2RS BGP Information Model", draft-hares-i2rs-bgp-im-00 (work in progress), July 2014.
- [I-D.hares-i2rs-info-model-service-topo]  
Hares, S., Wu, W., and X. Guan, "An Information model for service topology", draft-hares-i2rs-info-model-service-topo-00 (work in progress), February 2014.



- [I-D.ietf-i2rs-architecture]  
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-04 (work in progress), June 2014.
- [I-D.ietf-i2rs-rib-info-model]  
Bahadur, N., Folkes, R., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-03 (work in progress), May 2014.
- [I-D.krishnan-i2rs-large-flow-use-case]  
ramki, r., Ghanwani, A., Kini, S., McDysan, D., and D. Lopez, "Large Flow Use Cases for I2RS PBR and QoS", draft-krishnan-i2rs-large-flow-use-case-04 (work in progress), April 2014.
- [I-D.white-i2rs-use-case]  
White, R., Hares, S., and A. Retana, "Protocol Independent Use Cases for an Interface to the Routing System", draft-white-i2rs-use-case-05 (work in progress), June 2014.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3060] Moore, B., Ellessen, E., Strassner, J., and A. Westerinen, "Policy Core Information Model -- Version 1 Specification", RFC 3060, February 2001.
- [RFC3644] Snir, Y., Ramberg, Y., Strassner, J., Cohen, R., and B. Moore, "Policy Quality of Service (QoS) Information Model", RFC 3644, November 2003.
- [RFC5394] Bryskin, I., Papadimitriou, D., Berger, L., and J. Ash, "Policy-Enabled Path Computation Framework", RFC 5394, December 2008.
- [RFC5511] Farrel, A., "Routing Backus-Naur Form (RBNF): A Syntax Used to Form Encoding Rules in Various Routing Protocol Specifications", RFC 5511, April 2009.

Authors' Addresses

Susan Hares  
Huawei  
7453 Hickory Hill  
Saline, MI 48176  
USA

Email: shares@ndzh.com

Qin Wu  
Huawei  
101 Software Avenue, Yuhua District  
Nanjing, Jiangsu 210012  
China

Email: sunseawq@huawei.com

I2RS working group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 2, 2015

S. Hares  
Q. Wu  
M. Wang  
J. You  
Huawei  
January 29, 2015

An Information model for service topology  
draft-hares-i2rs-info-model-service-topo-03

## Abstract

As stated in [I.D-ietf-sfc-problem-statement], the service overlay is independent of the network topology and allows operators to use whatever overlay or underlay they prefer and to locate service nodes in the network as needed.

This document extends the general topology model concept defined in [I.D-medved-i2rs-topology-im] and focuses on defining information model for service topology.

## 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 2, 2015.

## Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

## Table of Contents

1. Introduction . . . . .	2
2. Conventions used in this document . . . . .	3
3. Service Topology Information Model . . . . .	3
3.1. Model Overview . . . . .	3
3.2. Abstract Topology Model: the Service-Topology Component . . . . .	3
3.3. Model Extension: Service Function Chain Topology Component . . . . .	7
3.4. Model Extension: Inventory datastore Component . . . . .	8
4. Security Considerations . . . . .	9
5. IANA Considerations . . . . .	9
6. References . . . . .	9
6.1. Normative References . . . . .	9
6.2. Informative References . . . . .	10
Authors' Addresses . . . . .	10

## 1. Introduction

Network topology information can be collected from network by using IGP or BGP-LS [I.D-draft-ietf-idr-ls-distribution]. Information model for network topology provided in [I.D-medved-i2rs-topology-im] is built based on such network topology information.

A service specific overlay utilized by Service chaining creates the service topology. The overlay creates a path between service function(SF) nodes. Service functions can be co-located on one SF Node or physically separated across several SF Nodes with each having one or more Service Functions. In either case, a service function may be running in its own virtualized system space or natively on the hosting system.

Within the service topology, an ordered set of Service functions will be invoked for each packet that belongs to a given flow for which a SFC will be applied. Adding new service function to SF Node in the topology is easily accomplished, and no underlying network changes are required. Furthermore, additional service Functions or Service Function instances, for redundancy or load distribution purpose, can be added or removed to the service topology as required.

As stated in [I.D-ietf-sfc-problem-statement], the service overlay is independent of the network topology and allows operators to use

whatever overlay or underlay they prefer and to locate service nodes in the network as needed.

This document extends the general topology model concept defined in [I.D-medved-i2rs-topology-im] and focuses on defining information model for service topology.

## 2. Conventions used in this document

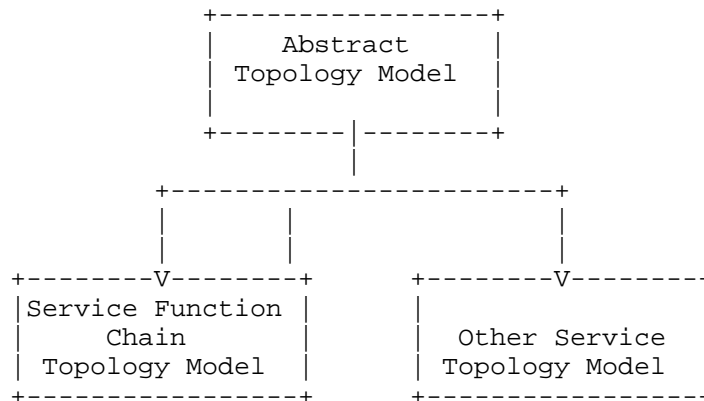
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [RFC2119].

## 3. Service Topology Information Model

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

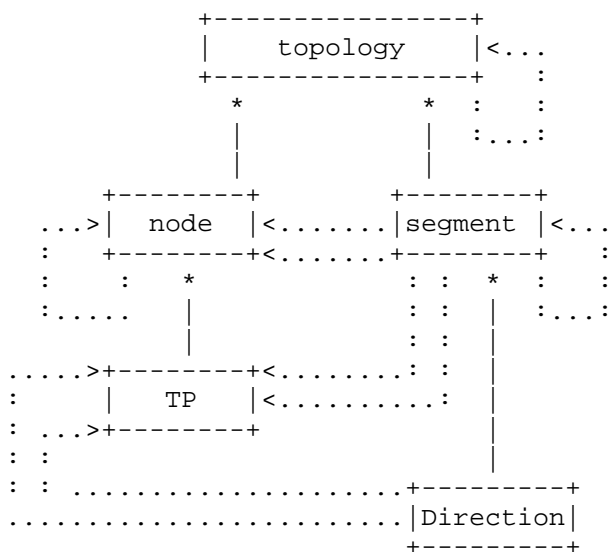
### 3.1. Model Overview

The abstract Topology Model contain a set of abstract nodes and a list of abstract links. An abstract link connects two abstract nodes. Service Function Chain Topo model and other service topo model can be augmented from the abstract topology model with topology specifics.



### 3.2. Abstract Topology Model: the Service-Topology Component

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



The basic information model works as follows: A service topology contains service nodes and segments. A segment connects two nodes (a source and a destination) and have direction, may be unidirectional or bidirectional. unidirectional is one where traffic is passed through any two service node or a set of service nodes in one forwarding direction only. Bidirectional is one where traffic is passed through any two service nodes or a set of service nodes in both forwarding directions. Each service node contains termination points. It occurs before or after other service node, therefore each node may have its upstream service node and/or downstream service node.

A service node may be dedicated to a tenant(e.g., an IPVPN customer), globally shared among tenants, or available to be assigned in whole or in part to a tenant or a set of tenants. Therefore service Nodes can map onto and be supported by other network elements in the underlying network, while Segment can map onto and be supported by other links in the underlying network,e.g., one segment can be mapped to two consecutive links stitching together. Service Topologies can map onto other, underlay topologies.

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

```
/* exterior definitions for service topology */
<service-topology> ::= (<topology>...)

/* Topology definitions */
```

```

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

<node-count> ::= INTEGER-32;
<topology-type> ::= (
                      (<netconf> [<netconf-topology-type>]) |
                      (<i2rs> [<i2rs-topology-type>])

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

<topology-extension> ::=
                      <netconf-topology-extension> |
                      ...

<segment> ::= <Segment_IDENTIFIER>
               <source>
               <destination>
               [<direction>]
               [<segment-extension> ]

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

<termination-point-reference> ::= <SF_NODE_IDENTIFIER>

<direction> ::= (<Unidirection>)|
                 (<Bidirection>)

<segment-extension> ::= <netconf-segment-extension> |
                        <i2rs-segment-extension> |
                        ...

<node> ::= <SF_NODE_IDENTIFIER>
           (<termination-point>...)
           [<NODE_TYPE>]
           [<NEXT-HOP>]
           [<node-extension>]

<termination-point> ::= <TERMINATION_POINT_IDENTIFIER>
                        [<supporting-termination-points>]
                        [<termination-point-extension>]
<supporting-termination-points> ::=
                        (<TERMINATION_POINT_IDENTIFIER>...)
< NODE-TYPE> ::=
                (<Classifier-Node>)|

```

```

        (<SF-Node>)|
        (<SFF-Node>)
    ...
<NEXT-HOP> ::= (<NODE_IDENTIFIER>...)
<node-extension> ::= <Classifier-extension> |
                    <SF-Node-extension> |
                    <SFF-Node-extension>
    ...

```

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

- o A service overlay can contain multiple topologies. Each topology is captured in its own list element, distinguished via a topology-id.
- o A topology has a certain type, such as NETCONF or I2RS. A topology can even have multiple types simultaneously. The type, or types, are captured in the list of "topology-type" components.
- o A topology contains segments and nodes, each captured in their own list.
- o A node has a node-id. This distinguishes the node from other nodes in the list. In addition, a node has a list of termination points, used to terminate segment. An examples of a termination point might be a physical or logical port or, more generally, an interface.
- o A segment is identified by a segment-identifier, uniquely identifying the portion of the network bounded by two service nodes within the topology. segment are point-to- point and has direction. The direction can be unidirectional or bidirectional. Accordingly, a segment contains a source and a destination. Both source and destination reference a corresponding node, as well as a termination point on that node.
- o The topology, node, segment and direction elements can be extended with topology-specific components (topology-extensions, node-extension, segment-extension and direction-extension respectively).

The topology model includes segment that are either bidirectional unidirectional. Service function chain path is analogue to linked list data structure and can be represented through a set of Ordered segments from source to destination. Each node in the service overlay may be located at different layer. The segment can be setup



to steer traffic through these specific service nodes at different layers or bypass some specific service nodes at different layers.

The topology model only supports point to point and does not support multipoint. Therefore Segments are terminated by a single termination point, not sets of termination points. Connections involving multihoming or segment aggregation schemes need to be modeled using multiple point-to-point segment, e.g., connection from service node A at lower layer to service node D at higher layer can comprise a segment 1 from service node A to service node B and segment 2 from service node B to service node C and segment 3 from service node C to service node D. By using segment aggregation, we can define a new segment from service A to service node D which is supported by segment 1, 2 and 3.

Unlike network topology collection, the service topology information may be not available from each SF by using IGP advertisement or BGP-LS northbound distribution since SF may be not located at network layer. However these SF at different layer may have affinity with one SF node (e.g., SF egress node or SF ingress node or SF enabled node), therefore service topology information associated with Service nodes can be collected using RESTCONF/NETCONF interface or I2RS interface for interrogation of a virtual device's state, statistics and configuration.

### 3.3. Model Extension: Service Function Chain Topology Component

```

<Classifier-extension> ::= <SFP>
                             <SFC-Policy>
                             <Matching-RULE>

<SFP> ::= <SF-List>
          <SFF-List>
<SF-Node-extension> ::= <SF-Node-Locator>
                        <Support-Context-Type>
                        <SF-Type>
                        <SF-Inventory-data>

<SF-type> ::=
    <firewall> |
    <loadbalancer> |
    <NAT44> |
    <NAT64> |
    <DPI>

<SFF-Node-extension> ::= <SFFN-address>
                        <SFFN-Virtual-Context>
                        <Attached-service-add>
                        <Customer-Support-List>
                        <Customer-Support-Resource-List>
                        <SFFN-VNTopo>
<SFFN-Virtual-Context> ::= <id>

```

#### 3.4. Model Extension: Inventory datastore Component

Inventory Data for service overlay can be obtained by using NETCONF or I2RS and share to PCE, ALTO server or other topology manager defined in [I.D-ietf-i2rs-architecture]. Information shared by them is defined as the component, "inventory database". This component defines a set of groupings with auxiliary information required and shared by those other components.

```
<SF-inventory-data> ::=
    <SF-capabilities>
    <SF-administrative-info>

<SF-capabilities> ::=
    (<supported-ACL-number >)|
    (<virtual-context-number >)|
    (<supported-packet-rate>)|
    (<supported-bandwidth>)

<SF-administrative-info> ::= =
    (<Packet-rate-utilization>)|
    (<Bandwidth-utilization-per-CoS>)|
    (<Packet-rate-utilization-per-Cos>)|
    (<Memory-utilization>)|
    (<available-memory>)|
    (<RIB-utilization-per-address-family>)|
    (<FIB-utilization-per-address-family>)|
    (<CPU-utilization>)|
    (<Available storage>)|
    (<Bandwidth-utilization>)|
    (<Flow-resource-utilization-per-flow-type>)
```

This module details inventory node attributes:

- o Inventory node attributes include SF-type, SF-capabilities and SF-administrative-info.

#### 4. Security Considerations

This document does not introduce any new security issues above those identified in [RFC5511].

#### 5. IANA Considerations

This draft includes no request to IANA.

#### 6. References

##### 6.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.
- [RFC5511] Farrel, A., "Routing Backus-Naur Form (RBNF): A Syntax Used to Form Encoding Rules in Various Routing Protocol Specifications", RFC 5511, April 2009.

## 6.2. Informative References

- [I.D-bitar-i2rs-service-chaining]  
Bitar, N., Heron, G., and L. Fang, "Interface to the Routing System (I2RS) for Service Chaining: Use Cases and Requirements", ID draft-bitar-i2rs-service-chaining-00, July 2013.
- [I.D-draft-ietf-idr-ls-distribution]  
Gredler, H., "North-Bound Distribution of Link-State and TE Information using BGP", ID draft-ietf-idr-ls-distribution-03, May 2013.
- [I.D-ietf-sfc-problem-statement]  
Quinn, P., "Service Function Chaining Problem Statement", ID draft-ietf-sfc-problem-statement-10, August 2014.
- [I.D-medved-i2rs-topology-im]  
Medved, J., Bahadur, N., Clemm, A., and H. Ananthakrishnan, "An Information Model for Network Topologies", ID draft-medved-i2rs-topology-im-01, October 2003.

## Authors' Addresses

Susan Hares  
Huawei  
7453 Hickory Hill  
Saline, MI 48176  
USA

Email: shares@ndzh.com

Qin Wu  
Huawei  
101 Software Avenue, Yuhua District  
Nanjing, Jiangsu 210012  
China

Email: sunseawq@huawei.com

Michael Wang  
Huawei  
101 Software Avenue, Yuhua District  
Nanjing, Jiangsu 210012  
China

Email: wangzitao@huawei.com

Jianjie You  
Huawei  
101 Software Avenue, Yuhua District  
Nanjing, Jiangsu 210012  
China

Email: youjianjie@huawei.com

I2RS working group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 10, 2015

S. Hares  
Huawei  
S. Brim  
Consultant  
N. Cam-Winget  
Cisco  
J. Halpern  
Ericcson  
D. Zhang  
Q. Wu  
Huawei  
A. Abro  
S. Asadullah  
Cisco  
J. Halpern  
Ericcson  
E. Yu  
Cisco  
February 6, 2015

I2RS Security Considerations  
draft-hares-i2rs-security-03

Abstract

This presents an expansion of the security architecture found in the i2rs architecture.

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 10, 2015.

## Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	2
2. Definitions . . . . .	3
3. Security Issues . . . . .	5
4. Security roles and Identities for the I2RS client and I2RS Agent . . . . .	6
5. I2RS Data Security . . . . .	7
5.1. Data Confidentiality Requirements . . . . .	8
5.2. Message Integrity Requirements . . . . .	8
6. Open Issues . . . . .	9
7. Acknowledgement . . . . .	9
8. IANA Considerations . . . . .	9
9. Security Considerations . . . . .	9
10. Informative References . . . . .	10
Authors' Addresses . . . . .	11

## 1. Introduction

The Interface to the Routing System (I2RS) provides read and write access to the information and state within the routing process and configuration process (as illustrated in the diagram in the architecture document within routing elements. The I2RS client [I-D.ietf-i2rs-architecture] interacts with one or more I2RS agents to collect information from network routing systems. This security architecture expands on the security issues involved in the I2RS protocol's message exchange between the I2RS client and the I2RS agent which are described in [I-D.ietf-i2rs-architecture]

## 2. Definitions

This document utilizes the definitions found in the following drafts: [RFC4949], and [I-D.ietf-i2rs-architecture]

Specifically, this document utilizes the following definitions:

### Authentication

[RFC4949] describes authentication as the process of verifying (i.e., establishing the truth of) an attribute value claimed by or for a system entity or system resource. Authentication has two steps: identify and verify.

### Data Confidentiality

[RFC4949] describes data confidentiality as having two properties: a) data is not disclosed to system entities unless they have been authorized to know, and b) data is not disclosed to unauthorized individuals, entities or processes. The key point is that confidentiality implies that the originator has the ability to authorize where the information goes. Confidentiality is important for both read and write scope of the data.

### Data confidentiality service

[RFC4949] also describes data confidentiality service as a security service that protects data against unauthorized disclosure. Please note that an operator can designate all people are authorized to view a piece of data which would mean a data confidentiality service would be essentially a null function.

### Data Privacy

[RFC4949] describes data privacy as a synonym for data confidentiality. This I2RS document will utilize data privacy as a synonym for data confidentiality.

### Mutual Authentication

[RFC4949] implies that mutual authentication exists between two interacting system entities. Mutual authentication in I2RS implies that both sides move from a state of mutual suspicion to mutually authenticated communication after each system has been identified and validated by its peer system

### Mutual Suspicion



[RFC4949] defines mutual suspicion as a state that exists between two interacting system entities in which neither entity can trust the other to function correctly with regard to some security requirement.

#### Role

[RFC4949] describes role as a job function or employment position to which people or other system entities may be assigned in a system. In the I2RS interface, the I2RS agent roles relate to the roles that the I2RS client is utilizing. In the I2RS interface, the I2RS client negotiation is over the client's ability to access resources made available through the agent's particular role.

#### Role-based Access control

[RFC4949] describes role-based access control as an identity-based access control wherein the system entities that are identified and controlled are functional positions in an organization or process. Within [RFC4949] five relationships are discussed: 1) administrators to assign identities to roles, 2) administrators to assign permissions to roles, 3) administrators to assign roles to roles, 4) users to select identities in sessions, and 5) users to select roles in sessions. This document discusses I2RS use of Roles as Scope+Access where scope is the portion of the routing tree, and access is permissions to read or write (or both). Figure 1 replicates [RFC4949] diagram on RBAC roles and assignments (page 254).

#### Role hierarchy or Permissions inheritance

[RFC4949] describes the hierarchy of roles and identities in role-based access control shown in Figure 1 and described above. I2RS will use role-based access control as defined above, and shown in Figure 2.

#### Role certificate

[RFC4949] describes a role certificate as an organizational certificate that is issued to a system entity that is a member of the set of users that have identities that are assigned to the same role.

#### Security audit trail

[RFC4949] (page 254) describes a security audit trail as a chronological record of system activities that is sufficient to enable the reconstruction and examination of the sequence



notification stream or publication stream as a pre-authorized read. This security consideration document examines the major points:

#### I2RS roles and identities

This section looks at how I2RS roles and identities created by [I-D.ietf-i2rs-architecture], how I2RS model derived from the security model of role-based access control matches the [I-D.ietf-i2rs-architecture], and how Identities and roles get distributed.

#### Data Security

The data security section looks at incidents when the I2RS data stream will need confidentiality and message integrity, transport security, how role-based access control of I2RS data impacts the I2RS Information Model and Data Model design, and light weight clients who work without confidentiality.

#### 4. Security roles and Identities for the I2RS client and I2RS Agent

All I2RS clients and I2RS agents MUST have at least one unique identifier that uniquely identifies each party. The I2RS protocol MUST utilize these identifiers for mutual identification of the client and agent. An I2RS agent, upon receiving an I2RS message from a client, must confirm that the client has a valid identity. The client, upon receiving an I2RS message from an agent, must confirm the I2RS identity.

Identity distribution and the loading of these identities into I2RS agent and I2RS Client occur outside the I2RS protocol. The I2RS protocol SHOULD assume some mechanism (IETF or private) in order to distribute or load identities and that the I2RS client/agent will load the identities prior to the I2RS protocol establishing a connection between I2RS client and I2RS agent.

Each Identity will be linked (via internal policy) to one role. The context of the I2RS client-agent communication is based on a role which may/may not require message confidentiality, message integrity protection, or replay attack protection.

The rigorous definition of a role in RBAC-based security is role is function associated with an activity (set of actions). The set of actions in I2RS performs is limited are read or write actions on a specific set of data in the data model. Therefore, we can express:

Role = routing tree + Read/Write/Read-Write

Role security for an agent involves pairing the identity to the role. The data store can read information either by write or an event stream.

Role security exists even if multiple transport connections are being used between the I2RS client and I2RS agent as the I2RS architecture [I-D.ietf-i2rs-architecture] states. These transport message streams may start/stop without affecting the existence of the client/agent data exchange. TCP supports a single stream of data. SCTP [RFC4960] provides security for multiple streams plus end-to-end transport of data.

I2RS clients may be used by multiple applications to configure routing via I2RS agents, receive status reports, turn on the I2RS audit stream, or turn on I2RS traceability. An I2RS client software could arrange to store multiple secure identities, and use a specific identity that only associates roles which only have Read access. This administrative design of identities and roles could insure a "status-only" application did gain write access. This administrative design is possible within I2RS architecture but not mandated.

Multiple identities provide some secondary level support for the application-client, but may grow the number of identities. The multiple identities per client could also be used for multiple levels of security for the data passed between an I2RS client and agent as either: a) confidential, b) authorized with message integrity protection, c) authorized without message integrity protection, and or d) no protection.

## 5. I2RS Data Security

I2RS data security involves determining of the I2RS client to I2RS agent data transfer needs to be confidential, or have message integrity, or support an end-to-end integrity (in the case of stacked clients). This section discuss the consideration of I2RS data security.

It is assumed that all I2RS data security mechanisms used for protecting the I2RS packets needs to be associated with proper key management solutions. A key management solution needs to guarantee that only the entities having sufficient privileges can get the keys to encrypt/decrypt the sensitive data. In addition, the key management mechanisms need to be able to update the keys before they have lost sufficient security strengths, without breaking the connection between the agents and clients.

The rules around what role is permitted to access and manipulate what information, combined with encryption to protect the data in transit

is intended to help ensure that data of any level of sensitivity is reasonably protected from being observed by those without permission to view it. In that case 'those' can refer to either other roles, sub-agents, or to attackers and assorted MITM monkeys.

#### 5.1. Data Confidentiality Requirements

In a critical infrastructure, certain data within routing elements is sensitive and R/W operations on such data must be controlled in order to protect its confidentiality. For example, most carriers do not want a router's configuration and data flow statistics known by hackers or their competitors. While carriers may share peering information, most carriers do not share configuration and traffic statistics. To achieve this, access control to sensitive data needs to be provided, and the confidentiality protection on such data during transportation needs to be enforced.

It is normal to protect the confidentiality of the sensitive data during transportation by encrypting them. Encryption obscures the data transported on the wire and protects them against eavesdropping attacks. Because the encryption itself cannot guarantee the integrity or freshness of data being transported, in practice, confidentiality protection is normally provided with integrity protection.

#### 5.2. Message Integrity Requirements

An integrity protection mechanism for I2RS should be able to ensure 1) the data being protected are not modified without detection during its transportation and 2) the data is actually from where it is expected to come from 3) the data is not repeated from some earlier interaction of the protocol. That is, when both confidentiality and integrity of data is properly protected, it is possible to ensure that encrypted data are not modified or replayed without detection.

As a part of integrity protection, the replay protection approaches provided for I2RS must consider both online and offline attackers, and have sufficient capability to deal with intra connection and inter-connection attacks. For instance, when using symmetric keys, sequence numbers which increase monotonically could be useful to help in distinguishing the replayed messages, under the assistance of signatures or MACs (dependent on what types of keys are applied). In addition, in the cases where only offline attacker is considered, random nonce could be effective.

## 6. Open Issues

The following are open issues for the I2RS WG to discuss:

### Unencrypted Message Exchanges

The I2RS Security discussion group believes that encrypting all the data messages is the best approach for security. Some I2RS WG discussion has indicated a desire for the the I2RS client-agent message exchanges to be unencrypted. The discussion group needs the I2RS WG members to provide more detail since a mixture of encrypted and unencrypted data will require more complexity in the Information Model (IM) and Data Model (DM).

### Transport requirements

The architecture provides the ability to have multiple transport sessions providing protocol and data communication between the I2RS Agent and the I2RS client. The discussion group proposed on mandatory secure transport. Should there be one mandatory secure transport protocol or multiple allowable protocols?

### Auditable Data Streams

Auditable data streams does not have a security consideration because I2RS is not inventing a new audit protocol as many protocols (syslog) are available to be used. Verifying audit stream data is outside the I2RS protocol, but those designing the IM and DMs with audit stream capability need to provide the appropriate hooks such as: on/off action, data selection, and protocol (for example syslog) that the I2RS Agent (or I2RS routing system) sends the audit data upon.

## 7. Acknowledgement

The authors would like to thank Wes George, Ahmed Abro, Qin Wu, Eric Yu, Alia Atlas, and Jeff Haas for their wonderful contributions to our discussion discussion.

## 8. IANA Considerations

This draft includes no request to IANA.

## 9. Security Considerations

This is a document about security architecture beyond the consideration for I2RS. Additional security definitions will be added in this section.

## 10. Informative References

- [I-D.clarke-i2rs-traceability]  
Clarke, J., Salgueiro, G., and C. Pignataro, "Interface to the Routing System (I2RS) Traceability: Framework and Information Model", draft-clarke-i2rs-traceability-02 (work in progress), June 2014.
- [I-D.hares-i2rs-info-model-policy]  
Hares, S. and W. Wu, "An Information Model for Basic Network Policy", draft-hares-i2rs-info-model-policy-03 (work in progress), July 2014.
- [I-D.ietf-i2rs-architecture]  
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-08 (work in progress), January 2015.
- [I-D.ietf-i2rs-problem-statement]  
Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Problem Statement", draft-ietf-i2rs-problem-statement-06 (work in progress), January 2015.
- [I-D.ietf-i2rs-rib-info-model]  
Bahadur, N., Folkes, R., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-05 (work in progress), January 2015.
- [I-D.ji-i2rs-usecases-ccne-service]  
Ji, X., Zhuang, S., Huang, T., and S. Hares, "I2RS Use Cases for Control of Forwarding Path by Central Control Network Element (CCNE)", draft-ji-i2rs-usecases-ccne-service-02 (work in progress), July 2014.
- [I-D.keyupate-i2rs-bgp-usecases]  
Patel, K., Fernando, R., Gredler, H., Amante, S., White, R., and S. Hares, "Use Cases for an Interface to BGP Protocol", draft-keyupate-i2rs-bgp-usecases-04 (work in progress), July 2014.
- [I-D.white-i2rs-use-case]  
White, R., Hares, S., and A. Retana, "Protocol Independent Use Cases for an Interface to the Routing System", draft-white-i2rs-use-case-06 (work in progress), July 2014.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

- [RFC4785] Blumenthal, U. and P. Goel, "Pre-Shared Key (PSK) Ciphersuites with NULL Encryption for Transport Layer Security (TLS)", RFC 4785, January 2007.
- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

Authors' Addresses

Susan Hares  
Huawei  
7453 Hickory Hill  
Saline, MI 48176  
USA

Email: shares@ndzh.com

Scott Brim  
Consultant

Email: scott.brim@gmail.com

Nancy Cam-Winget  
Cisco

Email: ncamwing@cisco.com

Joel Halpern  
Ericsson

Email: joel.halpern@ericsson.com

DaCheng Zhang  
Huawei

Email: zhangdacheng@huawei.com



Qin Wu  
Huawei

Email: bill.wu@huawei.com

Ahmed Abro  
Cisco

Email: aabro@cisco.com

Salman Asadullah  
Cisco

Email: sasad@cisco.com

Joel Halpern  
Ericcson

Email: joel.halpern@ericsson.com

Eric Yu  
Cisco

Email: eyu@cisco.com

i2rs  
Internet-Draft  
Intended status: Informational  
Expires: November 15, 2015

S. Hares  
Huawei  
M. Chen  
Huawei Technologies  
May 14, 2015

Summary of I2RS Use Case Requirements  
draft-hares-i2rs-usecase-reqs-summary-02

Abstract

The I2RS Working Group (WG) has described a set of use cases that the I2RS systems could fulfil. This document summarizes these use cases. It is designed to provide requirements that will aid the design of the I2RS architecture, Information Models, Data Models, Security, and protocols.

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 November 15, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Protocol Independent Use Case Requirements . . . . .	4
3. BGP Use Case Requirements . . . . .	6
4. IGP Use Cases . . . . .	8
5. CCNE Use Cases . . . . .	10
6. Topology Related Use Cases . . . . .	10
6.1. Virtual Connection Use Case Requirements . . . . .	11
6.2. Virtual Network Use Case Requirements . . . . .	11
6.3. Topology Use Case . . . . .	12
6.4. Virtual Topology Data Model . . . . .	17
6.5. Virtual Topology IP Data Model . . . . .	18
6.6. Virtual Topology Network Element . . . . .	19
7. Requirements from SFC Use Cases . . . . .	20
8. Requirements from Traffic Steering Use Cases . . . . .	21
9. Requirements from MPLS TE Networks Use Cases . . . . .	22
10. Requirements from MPLS LDP Networks Use Cases . . . . .	24
11. Requirements from Mobile Backhaul Ues Cases . . . . .	25
12. Requirements from Large Data Flows are . . . . .	27
13. Large Data Collection Systems . . . . .	28
14. CDNI . . . . .	30
15. IANA Considerations . . . . .	31
16. Security Considerations . . . . .	31
17. References . . . . .	31
17.1. Normative References . . . . .	31
17.2. Informative References . . . . .	31
Authors' Addresses . . . . .	34

## 1. Introduction

The Architecture for the Interface to the Routing System [I-D.ietf-i2rs-architecture] allows for a mechanism where the distributed control plane can be augmented by an outside control plane through an open, accessible interface. This document summarizes the use case requirements for the I2RS client-I2RS Agent exchange found in the following documents:

- o Protocol Independent described in [I-D.white-i2rs-use-case]
- o BGP described in [I-D.keyupate-i2rs-bgp-usecases]
- o IGP protocols as described in [draft-ietf-wu-i2rs-igp-usecases]

- o Control of Forwarding Path by Central Control Network Element (CCNE) [I-D.ji-i2rs-usecases-ccne-service]
- o Virtual Connections and Virtual Networks described in [I-D.hares-i2rs-use-case-vn-vc]
- o Topology use cases [I-D.amante-i2rs-topology-use-cases]
- o Topology requirements [I-D.medved-i2rs-topology-requirements]
- o Service chaining described in [I-D.bitar-i2rs-service-chaining]
- o Traffic Steering described in [I-D.chen-i2rs-ts-use-case]
- o MPLS TE Networks described in [I-D.huang-i2rs-mpls-te-usecases]
- o MPLS LDP Networks described in [I-D.chen-i2rs-mpls-ldp-usecases]
- o Mobile BackHaul Use cases described in [I-D.zhang-i2rs-mbb-usecases]
- o Large Flows use case described in [I-D.krishnan-i2rs-large-flow-use-case]
- o Large Data Collection Systems Use cases described in [I-D.swhyte-i2rs-data-collection-system]
- o CDNI requesting routing [I-D.shin-i2rs-usecases-cdni-request-routing]

Each group of use cases is presented in its own document. Each use case is labeled with an identifier TTT-REQ-nn where TTT represents the type of use case. The abbreviations for TTT are:

- o PI - Protocol Independent
- o BGP - BGP
- o IGP - IGP protocols
- o CCNE - CCNE control of forwarding path
- o VCoD - Virtual Connections on Demand
- o VNoD - Virtual Networks on Demand
- o Topo - Topology Information

- o VT-TMD - Virtual Topology: Topology Data Model
- o VT-TDM-IP - Virtual Topology: Topology Data Mode for IP/MPLS
- o SFC - Service Chaining requirements
- o TS - Traffic Steering
- o MPLS-LDP - MLPS Topologies supported by LDP
- o MPLS-TE - MPLS-TE topologies
- o MBH - Mobile Back-Haul
- o L-Flow - Large Flows
- o L-Data - Large Data Collection
- o CDNI - CDNI networks

Each use case is also augmented with a notation signifying whether it is in or out of scope with regard to the current I2RS charter:

- o IC: In charter
- o OC: Out of charter
- o NA: not applicable to I2RS protocol, agent, client or models. Usually related to specific client-side app requirements.
- o ??: indicates this item needs additional classification aid from the WG.

In some cases a specific draft may be out of charter, but (sub)components of it's requirement set may be in charter. In charter. As such, (IC|OC|NA) designations may appear at the draft level, at the requirement level, or at the sub requirement level. In instances where designations do not appear at more specific level, the designation at the parent level should be considered to be inherited.

## 2. Protocol Independent Use Case Requirements

This is a summary of the I2RS requirements found in the Protocol Independent Use Cases described in: [I-D.white-i2rs-use-case] (IC):

- o PI-REQ01 (IC): The ability to monitor the available routes installed in the RIB of each forwarding device, including near

real time notification of route installation and removal. This information must include the destination prefix (NLRI), a table identifier (if the forwarding device has multiple forwarding instances), the metric of the installed route, and an identifier indicating the installing process.

- o PI-REQ02 (IC): The ability to install source and destination based routes in the local RIB of each forwarding device. This must include the ability to supply the destination prefix (NLRI), the source prefix (NLRI), a table identifier (if the forwarding device has multiple forwarding instances), a route preference, a route metric, a next hop, an outbound interface, and a route process identifier.
- o PI-REQ03 (IC): The ability to install a route to a null destination, effectively filtering traffic to this destination.
- o PI-REQ04(??): The ability to interact with various policies configured on the forwarding devices, in order to inform the policies implemented by the dynamic routing processes. This interaction should be through existing configuration mechanisms, such as NETCONF, and should be recorded in the configuration of the local device so operators are aware of the full policy implemented in the network from the running configuration.
- o PI-REQ05 (OC): The ability to interact with traffic flow and other network traffic level measurement protocols and systems, in order to determine path performance, top talkers, and other information required to make an informed path decision based on locally configured policy.
- o PI-REQ06 (IC): The ability to install destination based routes in the local RIB of each forwarding device. This must include the ability to supply the destination prefix (NLRI), a table identifier (if the forwarding device has multiple forwarding instances), a route preference, a route metric, a next hop, an outbound interface, and a route process identifier.
- o PI-REQ07 (IC): The ability to read the local RIB of each forwarding device, including the destination prefix (NLRI), a table identifier (if the forwarding device has multiple forwarding instances), the metric of each installed route, a route preference, and an identifier indicating the installing process.
- o PI-REQ08 (IC): The ability to read the tables of other local protocol processes running on the device. This reading action should be supported through an import/export interface which can present the information in a consistent manner across all protocol

implementations, rather than using a protocol specific model for each type of available process.

- o PI-REQ09 (OC for some protocols): The ability to inject information directly into the local tables of other protocol processes running on the forwarding device. This injection should be supported through an import/export interface which can inject routing information in a consistent manner across all protocol implementations, rather than using a protocol specific model for each type of available process.
- o PI-REQ10 (OC): The ability to interact with policies and configurations on the forwarding devices using time based processing, either through timed auto-rollback or some other mechanism. This interaction should be through existing configuration mechanisms, such as NETCONF, and should be recorded in the configuration of the local device so operators are aware of the full policy implemented in the network from the running configuration.

### 3. BGP Use Case Requirements

This is a summary of the requirements listed in [I-D.keyupate-i2rs-bgp-usecases] are (IC):

- o BGP-REQ01 (IC): I2RS client/agent exchange SHOULD support the read, write and quick notification of status of the BGP peer operational state on each router within a given Autonomous System (AS). This operational status includes the quick notification of protocol events that proceed a destructive tear-down of BGP session
- o BGP-REQ02 (IC): I2RS client SHOULD be able to push BGP routes with custom cost communities to specific I2RS agents on BGP routers for insertion in specific BGP Peer(s) to aid Traffic engineering of data paths. These routes SHOULD be tracked by the I2RS Agent as specific BGP routes with customer cost communities. These routes (will/will not) installed via the RIB-Info.
- o BGP-REQ03 (IC): I2RS client SHOULD be able to track via read/notifications all Traffic engineering changes applied via I2RS agents to BGP route processes in all routers in a network.
- o BGP-REQ04 (IC): I2RS Agents SHOULD support identification of routers as BGP ASBRs, PE routers, and IBGP routers.

- o BGP-REQ05 (IC): I2RS client-agent SHOULD support writing traffic flow specifications to I2RS Agents that will install them in associated BGP ASBRs and the PE routers.
- o BGP-REQ06 (IC): I2RS Client SHOULD be able to track flow specifications installed within a IBGP Cloud within an AS via reads of BGP Flow Specification information in I2RS Agent, or via notifications from I2RS agent
- o BGP-REQ07 (IC): I2RS client-agent exchange SHOULD support the I2RS client being able to prioritize and control BGP's announcement of flow specifications after status information reading BGP ASBR and PE router's capacity. BGP ASBRs and PE routers functions within a router MAY forward traffic flow specifications received from EBGP speakers to I2RS agents, so the I2RS Agent SHOULD be able to send these flow specifications from EBGP sources to a client in response to a read or notification.
- o BGP-REQ08 (IC): I2RS Client SHOULD be able to read BGP route filter information from I2RS Agents associated with legacy BGP routers, and write filter information via the I2RS agent to be installed in BGP RR. The I2RS Agent SHOULD be able to install these routes in the BGP RR, and engage a BGP protocol action to push these routers to ASBR and PE routers.
- o BGP-REQ09 (IC): I2RS client(s) SHOULD be able to request the I2RS agent to read BGP routes with all BGP parameters that influence BGP best path decision, and write appropriate changes to the BGP Routes to BGP and to the RIB-Info in order to manipulate BGP routes
- o BGP-REQ10 (IC): I2RS client SHOULD be able instruct the I2RS agent(s) to notify the I2RS client when the BGP processes on an associated routing system observe a route change to a specific set of IP Prefixes and associated prefixes. Route changes include: 1) prefixes being announced or withdrawn, 2) prefixes being suppressed due to flap damping, or 3) prefixes using an alternate best-path for a given IP Prefix. The I2RS agent should be able to notify the client via publish or subscribe mechanism.
- o BGP-REQ11 (IC): I2RS client SHOULD be able to read BGP route information from BGP routers on routes in received but rejected from ADJ-RIB-IN due to policy, on routes installed in ADJ-RIB-IN, but not selected as best path, and on route not sent to IBGP peers (due to non-selection).
- o BGP-REQ12 (IC): I2RS client SHOULD be able to request the I2RS agent to read installed BGP Policies.



- o BGP-REQ13 (IC): I2RS client SHOULD be able to instruct the I2RS Agent to write BGP Policies into the running BGP protocols and into the BGP configurations.
- o BGP-REQ14 (IC): I2RS client-agent SHOULD be able to read BGP statistics associated with Peer, and to receive notifications when certain statistics have exceeded limits. An example of one of these protocol statistics is the max-prefix limit.
- o BGP-REQ15 (IC): The I2RS client via the I2RS agent MUST have the ability to read the loc-RIB-In BGP table that gets all the routes that the CE has provided to a PE router.
- o BGP-REQ16 (IC): The I2RS client via the I2RS agent MUST have the ability to install destination based routes in the local RIB of the PE devices. This must include the ability to supply the destination prefix (NLRI), a table identifier, a route preference, a route metric, a next-hop tunnel through which traffic would be carried
- o BGP-REQ17 (IC): The I2RS client via the I2RS agent SHOULD have the ability to read the loc-RIB-in BGP table to discover overlapping routes, and determine which may be safely marked for removal.
- o BGP-REQ18 (IC): The I2RS client via the I2RS Agent SHOULD have the ability to modify filtering rules and initiate a re-computation of the local BGP table through those policies to cause specific routes to be marked for removal at the outbound eBGP edge.

#### 4. IGP Use Cases

This is a summary of the requirements listed in (ietf-draft-wu-ir2s-igp-usecases-00.txt) (OC):

- o IGP-REQ-01 (OC): I2RS Client/Agent SHOULD Be able to read/write the the unique IGP identification for router within an AS (router-id, system-id, or others). I2RS agents may notify the I2RS client of the detection of another router with the same unique ID.
- o IGP-REQ-02 (OC): I2RS Client SHOULD BE able to aid in IGP table reduction by actively monitoring IGP tables and by allowing changes to the IGP configuration in order to partition the IGPS and place ABRs and ASBRs. The I2RS Client/Agent exchange must allow for a rapid cycle of querying of IGP topology information and downloading of a new protocol configuration or updating of IGP nexthops in RIBs and FIBs to rapidly switch to new temporary IGP topologies. These alternate topologies may be calculated by a

application attached to the i2rs client and updated to the i2rs agent, or determined at the i2rs agent.

- o IGP-REQ-03 (OC): I2RS protocol and models should support Loop-Free Alternative (LFAs) [RFC5286] deployments in in pure IP and MPLS/LDP networks to provide single-point-failure protection for unicast traffic. This includes the configuration, monitoring of LFA changes, and letting off-line pre-computed paths for LFA backup of all links and prefixes in the network and calculating the protection coverage and recognizing optimization to be downloaded to appropriate devices via the I2RS interface (Client-Agent). Again, it is important to have deployment of changes followed by real-time feedback.
- o IGP-REQ-04 (OC): The I2RS programmatic interface SHOULD allow the balancing of both ECMP traffic flows and end-to-end traffic flows in the IGP. The I2RS SHOULD support monitoring of the dynamic traffic flow in the network, and the query of the maximum capacity of the network. This include the I2RS client's transmission to the I2RS agent of updated configuration after an off-line optimization to either spread traffic (across ECMP pathways) or aggregation of traffic onto a single path so the rest of the devices may power off saving power (and money).
- o IGP-REQ-05 (OC): The I2RS interface (protocol and data models) SHOULD use the subscription mechanism to filter the topology changes to interested events and use the publish mechanism to control the pace these events are notified. This filtering should protect the I2RS Client or even applications who depend on topology data from being drowned by massive original events or duplicate events from different sources
- o IGP-REQ-06 (OC): Since IGP protocol is essential to the whole network, the I2RS Clients SHOULD monitor about the protocol's running status before forwarding is impacted. Performance data can be collected through collecting static configuration and observing dynamic status. Static data includes the number of instances, interfaces, nodes in the network and etc. Dynamic data includes adjacency status, the number of entries in link-state database and in the routing table, the calculation status, the overload status, the graceful switch-over status, and others
- o IGP-REQ-07 (OC): The I2RS interface (protocol and IMs) should support a mechanism where the I2RS Clients can subscribe to the I2RS Agent's notification of critical node IGP events. For example, link-state database or routing table is under the status of overflow or the overflow status is released, the calculation continues for a long time, the system is under graceful reboot.

- o IGP-REQ-08 (OC): The I2RS interface (protocol and IMs) should support the reporting of IGP statistic such as dropped packet statistics. These statistics will aid detection of network failures or security attacks.

## 5. CCNE Use Cases

The use cases in I2RS Use Cases for Control of the Forwarding Path by a Central Control Network Element (CCNE)

[I-D.ji-i2rs-usecases-ccne-service] indicate the following requirements for I2RS (OC):

- o CCNE-REQ-01 (IC): I2RS interface should support I2RS client running on a CCNE to be able to pull information from both the BGP RR and the PCE. This information can include: BGP topology information, BGP routes, BGP statistics, BGP Peer topologies, PCE topology information, and PCE state information. The I2RS Client's request for reading of the RR and PCE topology information needs to have timely and rapid response from the I2RS Agent.
- o CCNE-REQ-02 (IC for some constraints): I2RS client should be able to set resource constraints at the I2RS Agent, and receive status information on the setting of resource constraints.
- o CCNE-REQ-03 (IC for some constraints): I2RS interface should be able to set service goal value to CCNE.
- o CCNE-REQ-04 (OC): I2RS client should be able support information models that allow re-optimization traffic model at CCNE .
- o CCNE-REQ-05 (IC): I2RS client should be able to receive notification at the CCNE, and be able to send status to the I2RS agent.
- o CCNE-REQ-06 (NA): I2RS client should work in parallel with traditional network management or OAM protocols sent to the general NE.
- o CCNE-REQ-07 (NA): I2RS clients should be able to to be light weight enough to be able to support running on a variety of devices (routers, centralized servers, or devices doing both).

## 6. Topology Related Use Cases

This section describes Topology or Virtual Topology related requirements the I2RS interface (protocol and information model (IM) included in the following types of use cases:

- o Virtual Connections on Demand: VCoD-REQ
- o Virtual Networks on Demand: VNoD-REQ
- o Virtual Topology Information Topo-REQ
- o Virtual Topology Data Model: VT-TDM-REQ
- o Virtual Topology IP Data Model: VT-TDMIP-REQ
- o Virtual Topology Network Element: VT-NE-REQ (TMF-GEN-1)

#### 6.1. Virtual Connection Use Case Requirements

- o VCoD-REQ01 (OC): I2RS Agents SHOULD provide the ability to read the virtual network topology database for the technology supported. For optical, these are the optical connections and what node they connect to, and the topologies created. For MPLS, this is virtual circuit available, what nodes they connect to, and the network topologies created. For IP technologies, this could include the GRE tunnels, what interface it connects to, and the topologies created. For Ethernet circuits this should involve circuit type (e.g, point-to-point (p2p) or point-to-multipoint (p2mp)) and what nodes it can reach, and the topologies created.
- o VCoD-REQ02 (OC): I2RS Agent SHOULD provide the ability to influence the configuration of a virtual circuit in a node.
- o VCoD-REQ03 (OC): I2RS Agent SHOULD provide monitor and provide statistics on the virtual connection to the I2RS client via a Read request or status Notification. The I2RS client can then determine if the connection falls below a quality level the application has requested. If the I2RS client does determine the circuit is below the required quality, it could create another circuit. The I2RS may choose to create the second virtual circuit, transfer flows, and then break the first circuit.

#### 6.2. Virtual Network Use Case Requirements

The requirements for the Virtual Networks on Demand (VCoD) are:

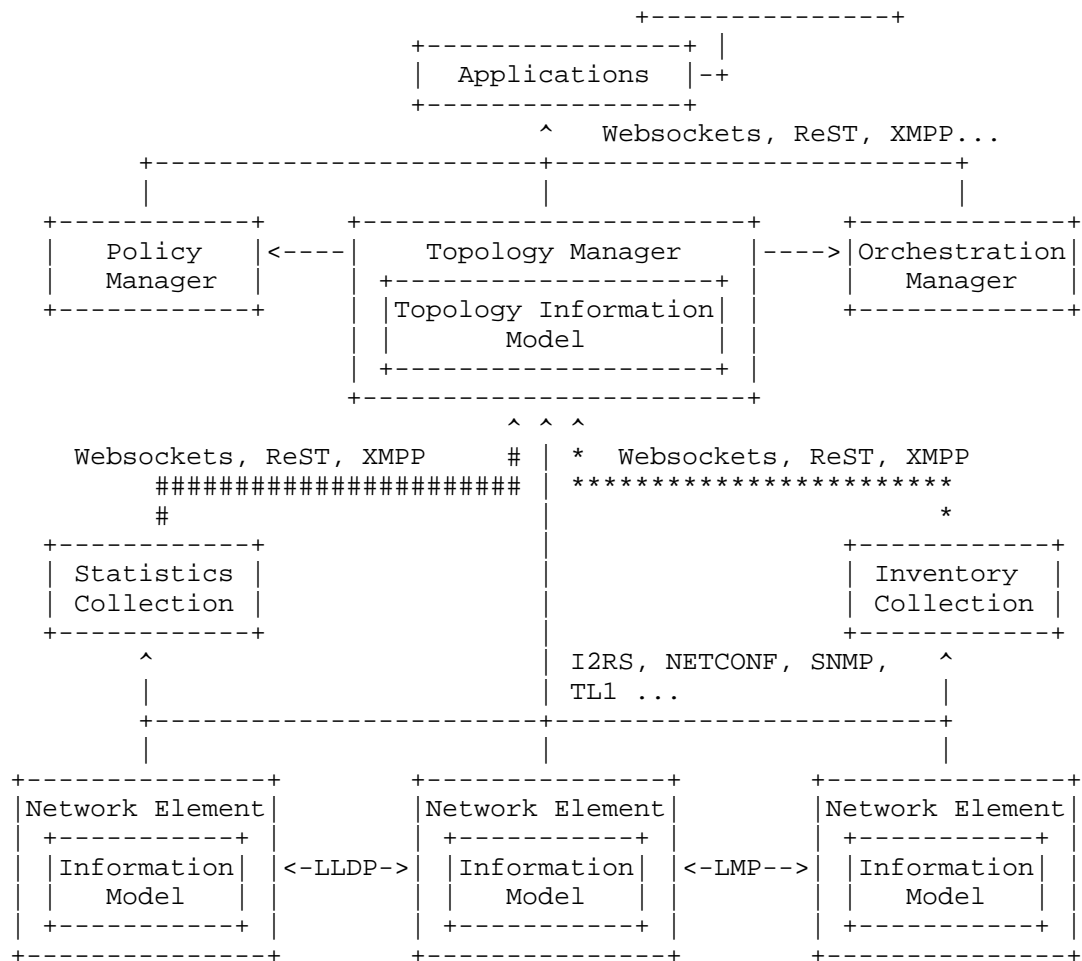
- o VT-VN-REQ01 (IC): I2RS Agents SHOULD provide the ability to read the virtual network topology database for the technology supported to determine nodes and connections. For optical, these are the optical connections and what node they connect to, and the topologies created. For MPLS, this is virtual circuit available, what nodes they connect to, and the network topologies created. For IP technologies, this could include the GRE tunnels, what

interface it connects to, and the topologies created. For Ethernet circuits this should involve circuit type (e.g, point-to-point (p2p) or point-to-multipoint (p2mp)) and what nodes it can reach, and the topologies created.

- o VNoD-REQ02 (IC): I2RS Agent SHOULD provide the ability to influence the configuration of a virtual circuit in a node.
- o VNoD-REQ03 (IC): I2RS Agent SHOULD provide monitor and provide statistics on the virtual connection to the I2RS client via a Read request or status Notification. The I2RS client can then determine if the connection falls below a quality level the application has requested. If the I2RS client does determine the circuit is below the required quality, it could create another circuit. The I2RS may choose to create the second virtual circuit, transfer flows, and then break the first circuit.
- o VNoD-REQ04 (IC): I2RS Agent SHOULD provide the ability to influence the configuration of a virtual network in a node.
- o VNoD-REQ05 (OC): I2RS Agent SHOULD provide the ability to report statistics on the network nodes and end-to-end traffic flows via read of status data or via notifications of status.
- o VNoD-REQ06 (IC): The I2RS protocol and RIB Informational Model (IM) must support logical tunnels of type MPLS as well as IP, GRE, VxLAN and GRE. Large Carrier networks utilize MPLS in a variety of forms (LDP, static MPLS TE, or dynamic TE LSPS created by RSVP-TE or CR-LDP).
- o VNoD-REQ07 (IC): I2RS SHOULD support Informational Models and features to allow MPLS technologies to create Hub-spoke topology and service routing in networks in Carriers, Enterprise, and Data Centers.
- o VNoD-REQ08 (IC): I2RS protocols, Information Models, and Data Models must be able to support Carriers using these MPLS technologies to support networks for Mobile BackHaul, on-demand MPLS overlays, and on-demand video conferencing networkings.

### 6.3. Topology Use Case

The requirements in [I-D.amante-i2rs-topology-use-cases] topology use cases focus around the architecture of topology manager, orchestration manager, and policy in the figure below (IC):



- o Topo-REQ-01 (IC): The Topology Manager Should be able to collect topological information via the I2RS Client-Exchange exchange from a variety of sources in a normalized topological model. These sources can be:
  - \* Live Layer IGP IGPs with information about the active topology such as the LSDB database or IGP updates,
  - \* The I2RS must enable the inventory system information to query for information about network components which are not visible to active L3. These systems can be active or simply invisible to the L3. Examples of this are L2 Ethernet switches or ROADMS.

- \* Statistic Collection systems that provide traffic information, such as traffic demands or link utilizations.

(from section 3.2)

- o Topo-REQ-02 (OC): Topology information is provided from Clients to high-layer applications via a northbound interface (such as ReST, Websockets, or XMPP).
- o Topo-REQ-03 (IC): Topology Manager should be able to collect and keep current topology information for multiple layers of the network: Transport, Ethernet and IP/MPLS, as well as information for multiple Layer 3 IGP areas and multiple Autonomous Systems (ASes). This information must contain cross-layer unerlying Shared Risk Link Groups (SRLG) within transport or Ethernet layers. (from section 3.2)
- o Topo-REQ-04 (OC): Topology manager be able to use I2RS Client-Agent protocol to to collect dynamic inventory information from network elements. An example of these protocols are the Link Layer discovery protocols (LLDP, LMP, etc.) which automatically identify remote nodes and ports. (from section 3.2)
- o Topo-REQ-05 (IC): I2RS Should enable the Policy manager to query and store the following types of policies:
  - \* Policies that contain Logical identifier Numbering in order to correlate IP Prefixes to
    - + link based on link type (P-P, P-PE, or PE-CE),
    - + IGP Area
    - + L2 VLAN assignments
  - \* Routing Configuration policies that correlate:
    - + OSPF area/ISIS Net-ID to Node (type)
    - + BGP node related policies (aggregation routes at node, max-prefix (per node), or AFI/SAFI per node
    - + Security policies - with ACLs or rate-limits
    - + Network Component access policies (for management

(from section 3.3)

- o Topo-REQ-06 (OC): I2RS should enable a orchestration manager attached to an I2RS client to communicate with I2RS agents into order to stitch together End-to-end services for network bandwidth optimization, load balancing, and Class-of-Service with point services (Firewall or NAT) within the end-to-end service). The orchestration manager should also be able to immediately schedule any of these resources via the I2RS-Client I2RS agent exchange. (from section 3.4)
- o Topp-REQ-07 (OC): The I2RS exchange should enable a statistics collector to collect statistics from the routing function of the network nodes and archive and aggregate the statistics into a statistics warehouse. Statistics must be given and stored in an normalized form. Metadata must be stored with the statistics. (from section 4.1.1.2) (Editor: there is some suggestion of periodic reports)
- o Topo-REQ-08 (IC): I2RS Client-I2RS agent exchange must be provide enough interoperability that the Topology manager, Policy manager, and inventory systems can be available from different vendors
- o Topo-REQ-09 (IC): TE tunnels must be able to be created by the exchange between the I2RS client and the I2RS agent. (from section 4.1.1)
- o Topo-Req-10 (NA): I2RS must provide a common and up-to-date normalized view of the topologies that that support security auditing, and IP/MPLS Provisioning (L2/L3) which includes:
  - \* Identifying Service PE's in all markets/cities where the customer has identified they want service,
  - \* Identifying one or more existing 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),
  - \* Obtain via query/notification the available capacity on Services PE in both the PE-CE access interface and its uplinks to terminate the tail circuit
  - \* Providing the context in I2RS for an iterative query mechanism needed by I2RS client attached to the the Topology to narrow down the scope of resources to the set of Services PEs with the appropriate uplink bandwidth and access circuit capability plus capacity to realize the requested VPN service.(from section 4.1.2)



- o Topo-REQ-11 (NA): The VPN application attached to the I2RS client should be able to hand the I2RS Client a candidate list of Service PE's and associated access circuits to set up a Customer's VPN service into the network. (from section 4.1.3) [Editor's note This request shares requirements with VCoD-REQ-01.]
- o Topo-REQ-12 (NA): The Topology Manager associated with the I2RS client must be able to use the normalized view of the network to set up additional queries (or notification publications) to provide an accurate and comprehensive picture in order a) diagnose faults/failures, and b) augment the network with additional services, and c) provide network topology maps for different purposes. (from section 4.1.3)
- o Topo-REQ-13 (IC): The I2RS client-agent exchange and informational models should support a Virtual Network Topology (VNT) comprise of one or more LSPs and lower layer resources. The VNT of MPLS must be able to link lower layer resources with the higher layer, and present a normalized form to the PCE as defined [RFC5623].
- o Topo-REQ-14 (OC): The I2RS client-agent protocol and models should support the use of a PCE to compute MPLS-TE paths within an "domain" (IGP area), or across multiple "domains" (multi-area AS, multiple ASes) as specified in [RFC4655]. This means the PCE Informational model should support:
  - \* enhanced computation in the single IGP domain
  - \* cross-AS path computation based on the multiple entrance of exit points from an AS,
  - \* linking multiple PEs in multiple domains together, and
  - \* synchronization of TED associated with the PCE to the topology manager (via I2RS client/messages), and
  - \* sending read/writes to the head-end-nodes(section 4.3)
- o Topo-REQ-15 (OC): the I2RS protocol and Information models should support the ALTO ([RFC5693]) generation of abstract network topology models and the APIs it support over web-service API. The ALTO abstract network topology comes in two forms: Network Map (based prefix-to PID mapping), and Cost map. The ALTO map is automatically generated from BGP and IGP data which the ALTO server queries from the network and makes available to applications via web-service API. (from section 4.4)

#### 6.4. Virtual Topology Data Model

The [I-D.medved-i2rs-topology-requirements] specifies the following Topology Data Model requirements (IC):

VT-TDM-REQ1 (IC): The topology data model MAY be able to describe topology and characteristics of the following layers:

- \* Optical DWDM (optional) (OC),
- \* Optical OTN (optional) (OC),
- \* L2 (Aggregated links, L2 topologies) (IC),
- \* IP/MPLS (IC),
- \* VPNs (IC), and
- \* Services (such as cloud services, or CDNs).

VT-TDM-REQ2 (IC): The topology data model MUST support multiple Autonomous System deployments.

VT-TDM-REQ3 (IC): The I2RS topology data model must support include topology information from multiple Administrative Domains or multiple elements into a single common format.

VT-TDM-REQ4 (IC): The I2RS topology data model MUST be able to convey enough information so that an I2RS client can correlate topologies in different layers and multiple Autonomous Systems.

VT-TDM-REQ5 (NA): The topology data model MUST support multi-layer group of elements as a means of coalescing different SFF Nodes and links into a network layers from various layers. For example, links with IPv4 addresses might represent Layer 3 of the network topology while links with Ethernet MAC addresses might represent Layer 2.

VT-TDM-REQ6 (IC): The topology model should allow association between components of different layers. For example, Layer 2 port may have several IPv4/IPv6 interfaces. The Layer-2 port and the IPv4/IPv6 interfaces would have an association.

VT-TDM-REQ7 (NA): The topology model MUST represent both inactive and active topologies in the topology Data base. Inactive topologies may include new line cards, ports in down state, etc.

VT-TMF-DM-REQ8 (NA): The topology data model MUST be hierarchical and MUST support summarization of sub-topologies. Topology summarization and creation of abstract topologies can be provided by either by the application associated with the I2RS client, or by the I2RS Agent prior to transmission to the I2RS client.

VT-TDM-REQ9 (IC): The topology data model MUST be able to describe abstract topologies. Abstract topologies can contain real and abstract nodes and real and abstract links. An abstract topology MAY be used by a provider to describe characteristics of a transit network (bandwidth, delay, protection, etc.)

VT-TDM-REQ10 (OC): The topology data model MUST support dynamic data, such as link and node utilizations (perhaps as optional attributes).

VT-TDM-REQ11a (??): The topology data model MUST allow I2RS client-agent to be able to identify and query for the path between two nodes.

VT-TDM-REQ11b (OC): The topology data model should support the I2RS Client requesting the I2RS Agent to trace the path at all network layers that participate in the delivery of packets between two nodes. This trace MAY involve either an I2RS Agent information trace or the I2RS Agent requesting the routing function trace the path at multiple levels (L3/L2.5/L2/L1)

VT-TDM-REQ12 (IC): The topology data model MUST support multiple BGP Autonomous Systems and multiple IGP areas. Support for multiple administrative domains is for further study.

VT-TDM-REQ13 (IC): The topology data model MUST be human-friendly, i.e. not SNMP MIBs, but something much more analogous to YANG models.

VT-TDM-REQ14 (IC): The data model SHOULD support topology abstraction, allowing clients that consume topology information in a constrained manner. For example, a client wishing to view only interfaces and nodes present in a sub-graph of the Layer 3 topology should be able to specify an interest in this subset of information rather than having to read out and parse through the entire set of links and nodes.

## 6.5. Virtual Topology IP Data Model

The [I-D.medved-i2rs-topology-requirements] specifies the following requirements for the Virtual Topology IP Data Model's IP/MPLS links and topologies (IC):

- o VT-TDM-IP-REQ1 (IC): The I2RS topology data model for the IP/MPLS layer MUST support both link topology and prefixes,
- o VT-TDM-IP-REQ2 (IC): The I2RS agent may import topology information from the routing processes, IGP process, BGP-LS information, or management processes.
- o TM-DM-IP-REQ3 (IC): The I2RS SFC Data model must support links that are IP/MPLS with the following attributes:
  - \* local and Remote anchor node IDs (Router ID, AS#, Area ID, MT topology),
  - \* metrics,
  - \* admin group,
  - \* max bandwidth links
  - \* unreserved/utilized bandwidth
  - \* link-protection type
  - \* MPLS protocol mask
  - \* link prefix
  - \* link characteristics (BW, Delay, error rate)
  - \* Link Description, and
  - \* Link-specific timers (Hello and Holddown).

#### 6.6. Virtual Topology Network Element

The [I-D.medved-i2rs-topology-requirements] specifies the following requirements (IC):

- o VT-NE-01 (IC): Each network element should contain an inventory data base which should be a definitive source of information with respect to the physical HW and Logical, logically significant identifiers (E.g. VLANs). The I2RS client should be able to import data from this DB into the I2RS Node IM or SFC IM.
- o VT-NE-02 (IC): The inventory DB of the network element should be augmented with the physical properties associated with the ports/interfaces that are directly connected to the device (BW, media

type). The I2RS client should be able to import data from this augmented DB into the I2RS Node IM or SFC IM.

- o NE-3 (NA): The I2RS client may write information into the NE inventory data base via the Network-element Data Model that the network element may not be able to learn on its own. This information may include the physical location (address), rack/bay information.

## 7. Requirements from SFC Use Cases

The SFC use case document in [I-D.bitar-i2rs-service-chaining] suggests that the following requirements (OC):

SFC-Use-REQ01 (IC):Address

has the following address requirements:

- \* IP address
- \* service-node tuple (service node IP address, Host system address)
- \* host-node tuple (hosting system IP-address, system internal identifier)

SFC-Use-REQ02 (IC):Supported Service Types

SHOULD include: NAT, IP Firewall, Load balancer, DPI, and others

SFC-Use-REQ03 (IC):Virtual contexts

SHOULD include:

- \* Maximum Number of virtual contexts supported
- \* Current number of virtual contexts in use
- \* Number of virtual contexts available
- \* Supported Context (VRF)

SFC-Use-REQ04 (IC): Customers currently on node

SFC-Use-REQ05 (IC): Customer Support Table (per customer ID)

- \* Customer-id

- \* List of supported Virtual Contexts

SFC-Use-REQ06 (OC): Service Resource table

which includes:

- \* index: Comprised of service node, virtual context, service type
- \* service bandwidth capacity
- \* supported packet rate (packets/second)
- \* supported bandwidth (kps)
- \* IP Forwarding support: specified as routing-instance(s), RIBs, Address-families supported
- \* Maximum RIB-size (WG Note: problematic)
- \* Maximum Forward Data Base size (WG Note: problematic)
- \* Maximum Number of 64 bit statistics counters for policy accounting
- \* Maximum number of supported flows for services (WG Note: problematic)

SFC-Use-REQ07 (IC): Virtual Network Topology (VNT)

which includes:

- \* number of access points to which service topology applies
- \* topology of access points

## 8. Requirements from Traffic Steering Use Cases

The requirements from the Traffic Steering use case described in [I-D.chen-i2rs-ts-use-case] are (OC):

- o TS-REQ01 (IC): The I2RS Client-Agent must be able to collect the topology (especially the exit links) and the traffic load of each link;
- o TS-REQ02 (IC): The I2RS Client-Agent must be able to read the local rib of each DC/Metro gateway and the policies deployed on each gateway;

- o TS-REQ03 (IC): The I2RS Client-Agent must be able to add or delete or modify the relevant rib items and relevant policies to steer the traffic as expected; and adjust traffic placement.
- o TS-REQ-04 (IC): The I2RS Client-Agent must have the ability to collect the LSP information either from the PCE or directly from network devices;
- o TS-REQ-05 (OC): The I2RS Client-Agent must have the ability to collect the traffic matrix of the network, this is used to help the I2RS client to determine how to adjust the traffic placement;
- o TS-REQ-06 (IC): The I2RS Client-Agent must have the ability to read the rib information and relevant policies of each network node;
- o TS-REQ-07 (OC): collect the topology and segment information needed to help the I2RS client to compute the end-to-end path;
- o TS-REQ-08 (OC): read rib (especially the segment routing rib) information;
- o TS-REQ-09 (??): add/delete/modify the segment rib, this finally determines how the traffic is forwarded.

#### 9. Requirements from MPLS TE Networks Use Cases

These are the requirements from the Traffic Steering use case described in [I-D.huang-i2rs-mpls-te-usecases] (OC):

- o MPLS-TE-REQ-01 (OC): Network programming software managing the static CR-LSP devices may incorporate an I2RS Client along with a path calculation entity, a label management entity, and a bandwidth management entity. The I2RS Client should be able to communicate the static configuration to the network nodes, and monitor the status of the CR-LSPs.
- o MPLS-TE-REQ-02 (OC): The I2Client should be able to synchronously send the configuration for all of the network nodes from egress node to ingress node via the I2RS Agents attached to each node, and be able to delay the final ingress node configuration until all the I2RS AGents on all other nodes toward the egress have denoted a successful path set-up.
- o MPLS-TE-REQ-03 (OC): MPLS TE defines abundant constraints such as explicit path, bandwidth, affinity, SRLG, priority, hop limit, and others. The I2RS Client Agent exchange should be able to signal concurrent local path calculation could obtain an optimized result

and allow more services to be held in a TE network. The I2RS Agent should be able to trigger a global concurrent re-optimization at a specific time on multiple nodes by communicating with each node's I2RS agent.

- o MPLS-TE-REQ-04 (NA): The I2RS client should be able to manually calculate a re-optimization of the the MPLS TE network and send the new constraints including the calculated path to each node via the I2RS agent with an indication to re-signal the TE LSPs with make-before-break method.
- o MPLS-TE-REQ-05 (OC): With I2RS, the node's I2RS agent should be able to send to an I2RS client a status notification that not enough resources exist for a back up LSP and TE tunnel. Upon receiving this notification the I2RS client should be able to trigger concurrent calculation for the failed path calculation of the backup LSP or TE tunnel and send the updated paths to I2RS agents with a command to re-signal the TE LSPS with make-before-break Method.
- o MPLS-TE-REQ-06 (NA): With I2RS, upon receipt the failure notification from an I2RS Agent, the I2RS client would create a global concurrent optimization to handle the failure event. This would occur by the I2RS client signalling the I2RS agents on all nodes to: a) trigger a new concurrent calculation of the backup LSP or TE tunnel via failed path calculation, and b) re-signal updates to the TE LSPs process with a make-before-break method.
- o MPLS-TE-REQ-07 (NA): Upon receiving a signal an upgrade event signal (from operator), the I2RS client could calculate another path for the affected TE tunnels to deviate traffic away from the resource being upgraded, and then send the request to I2RS agents on the appropriate nodes to move the traffic. After the upgrade completes, the I2RS client can simply remove I2RS configurations causing the traffic to revert to the original path. Or, the I2RS can re-optimize the TE tunnels for another pathways (E.g. as a part of a sequence of upgrades).
- o MPLS-TE-REQ-08 (OC): I2RS agents can notify I2RS Clients of impending or existing MPLS TE overload conditions that might cause TE LSP rejections. This overload conditions include: due to CPU, memory, LSP label space, or LSP numbers.
- o MPLS-TE-REQ-09 (IC): Automatic bandwidth adjustment applications can also be linked to the I2RS clients need to monitor the traffic on TE tunnels in order to provide traffic analysis. The I2RS client should be able to read the TE Tunnel topology and the bandwidth analysis in order to automatically calculate a new path for the TE



tunnel if it is needed. The I2RS Client also needs to be able to the I2RS agents in the nodes to install the new TE Tunnels with the make-before-break option.

- o MPLS-TE-REQ-10 (IC): With I2RS, the node failure or link failure can be part of the notification stream sent by an I2RS Agent to an I2RS Client on a centralized server gathering information.
- o MPLS-TE-REQ-11 (IC): The I2RS client can notify the I2RS agents on specific nodes (or devices) to re-signal TE LSPs one by one if there is a resource dependency.
- o MPLS-TE-REQ-12 (IC): The I2RS Client can gather the TE LSPs' state from I2RS Agents on all nodes in order to coordinate such handling of LSP resources.
- o MPLS-TE-REQ-13 (OC): The I2RS Clients collecting information from I2RS Agents can be arranged in a hierarchy to provide scaling of collections. An application hosting an I2RS client collecting information from I2RS Agents on nodes can have an I2RS Agent that reports combined information to a single location.

#### 10. Requirements from MPLS LDP Networks Use Cases

These are the I2RS requirements for the MPLS LDP use case described in [I-D.chen-i2rs-mpls-ldp-usecases]:

- o MPLS-LDP-REQ-01 (IC): The I2RS Client-agent exchange should allow the distribution of the configuration for PWE3, MPLS LDP and associated protocols to be distributed from a central location where the global PWE3 provisioning information could be stored. The I2RS Client-Agent exchange should also be able to push the configuration of the local LDP LSR ID and peer addresses to set up the targeted session to the pseudowire endpoints.
- o MPLS-LDP-REQ-02 (IC): When an the end-user wants to disable IPoMPLS (IP over MPLS) application on a L2VPN/PW Targeted LDP session, the I2RS Client-I2RS agent should be able to set type of application over the established LDP session. In this way LDP speaker can only advertise to its peer the application data which the user is interested in.
- o MPLS-LDP-REQ-03 (OC): The I2RS Agent notifications should allow an I2RS client to subscribe to a stream of state changes regarding the LDP sessions or LDP LSPs from the I2RS Agent. Specifically it is important that LDP session is tract for sessions state coming up or going down. The I2RS Client-I2RS Agent exchange should allow additional queries to the AAgent to determine a) why the

service is invalid, b) calculating whether an alternate path should be switched to, and c) determining how to switch to other links or nodes in order to recover from the link failure or node failure.

- o MPLS-LDP-REQ04 (IC): The I2RS interface provides way to monitor and control the limited resources on these access devices. The I2RS client should be able to instruct the I2RS agent in each of these devices to set the maximum number of LDP LSPs in each device prior to enabling LDP on the devices. The I2RS client should also be able to enable a notification service on each device with a warning threshold. Once the number of LDP LSPs reaches the threshold, the I2RS agent will send a notification message to the I2RS client. Often the I2RS client will be associated a network management agent that can determine what next steps need to be done based on policy or operator input.

#### 11. Requirements from Mobile Backhaul Use Cases

Mobile BackHaul Use cases described in [draft-ietf-zhang-mbb-usecases-01] are:

- o MBH-REQ-01 (OC): The I2RS client-agent communication can distribute position-critical changes to IGP nodes using this global knowledge to quicken changes to support traffic during failures or traffic overloads. To enable this feature, the I2RS Clients-Agent communication needs to pass information on which IGP process or Level or Area the given node and links belong to.
- o MBH-REQ-02 (OC): I2RS must allow operators to use of I2RS clients to distribute time-critical changes in configuration to I2RS agents associated with each routing node. This feature will simplify and automate configuration and monitoring of a mobile backhaul network to allow it to readily adapt to changing network sizes (and scales) and radio applications.
- o MBB-REQ-03 (OC): I2RS Clients-Agent communication needs to pass information on:
  - \* T-LDP configurations and status;
  - \* BGP peer configurations, peer topologies and status;
  - \* BGP-based LSP topologies and status;
  - \* Reset VPN topologies, and per node configurations;

- o MBB-REQ04 (IC): Route policy enforcement in mobile backhaul networks needs to be more dynamic and flexible than the current methods take hours (or even days) to configure route policy across a network. The I2RS interface must provide a programmatic way to configure (both policy and device) and monitor thousands of devices individually whose configuration is based on the devices role (such as ASRSs in one AS, ASBRs between ASs and other service-touch nodes).
- o MBB-REQ-05 (NA): I2RS clients should be able to contact I2RS agents on nodes to query role-based information from the network status. After collecting the status, the I2RS client can develop the BGP policies based on role information and push the BGP policies to the I2RS agents that would load the alternate policies into the network device. The I2RS Agents loading the alternate policies could then send status back to the I2RS Client.
- o MBH-REQ06 (??): I2RS clients can provide centralized control of many network devices via the I2RS Client-Agent communication. The I2RS programmatic interface can automate the collection and analysis of each device's capability so that the centralized I2RS client could calculate the optimal LSP path and distribute the configuration to individual devices. Automation of the collection of device capability should be available as query, notification, or a published stream.
- o MBH-REQ07 (NA): While the I2RS RIB Information Model [\[\[I-D.ietf-i2rs-rib-info-model\]\]](#) provides for routes with tunnels or MPLS LSP, the features defined in this model are not sufficient to configure both types of LSPs needed for the VPN technology in mobile backhaul networks. Additional I2RS Informational models need to be created to support these features.
- o MBH-REQ08 (NA): The hierarchical protection architecture in mobile backhaul network offer high network reliability and more flexibility to meet the various needs of the tunnels and services. The I2RS interface in this use case is needed to automate the configuration and monitoring so that tunnel protection and service protection interwork in a flexible and reliable manner.
- o MBB-REQ09 (OC): The I2RS architecture (client-agent) should allow the two features for network monitoring naturally in its basic modes:
  - \* allow a combination of multi-layer network monitor tools with exact detection parameters to be configured on the network device

- \* Facilitate the reporting the detection result as notification or publication stream

It is important the result of these features allow the outages and traffic congestion or discards to be detected real-time with I2RS Client(s) in each node, and the detection result will be reported to the I2RS agents to get the exact status of the network.

## 12. Requirements from Large Data Flows are

Each of these requirements has been given an ID number of L-Flow-nn for ease of reference.

The requirements from the Large Data Flows use case described in [I-D.krishnan-i2rs-large-flow-use-case] are (IC):

L-Flow-REQ-01 (IC): For redirecting large flows to a specific component, a PBR entry should be programmable for the flow with its nexthop that identifies the specific LAG or ECMP component.

L-Flow-REQ-02 (IC): For adjusting the weights used to distribute traffic across components of the LAG or ECMP, I2RS should provide a programmable mechanism should be provided that identifies ECMP entries and is able to associate weights that can be programmed for each of the components. To do this in a scalable fashion, it would be useful to have the notion of an ECMP nexthop that is used by multiple routes

L-Flow-REQ-03 (IC): The I2RS interface (protocol/IMs) should allow for a globally optimal path is programmed in the IP network using hop-by-hop PBR rules. These PBR rules may include:

- \* Being able to adjust the weights of the ECMP table for different nexthops should be adjusted to factor the large flows
- \* Being able to address an ECMP group, so that all routes sharing an ECMP group are addressed together.
- \* the ability to program PBR entries at the edge LSR, and
- \* the ability to program new LSPs in the network.

L-Flow-REQ-04 (OC): The I2RS protocol should be able to invoke the link aggregation IEEE 802.1AX Marker Protocol via the I2RS protocol. This is useful during a period of rebalancing occurs before flows are moved.

L-Flow-REQ-05 (IC): The I2rs protocol should allow Quality of Service (QoS) actions such as rate-limiting, re-marking, or discarding can be performed on the flows based on configured policies and nexthop redirection actions to be programmed, and to be programmed independently of each other.

L-Flow-REQ-06 (IC): Once a large flow has been detected, I2RS must be used to modify the forwarding tables in the router to:

- \* In the case of large flow load balancing, be able to redirecting the large flow to a particular member with the LAG or ECMP group and readjusting the weights of the other members to account for the large flow
- \* In the case of DDoS mitigation, the action involves rate limiting, remarking or potentially discarding the large flow in question.

### 13. Large Data Collection Systems

The requirements from the Large Data Collection Systems Use cases described in [draft-swhyte-i2rs-data-collection-system] are (OC):

L-Data-REQ-01 (OC): I2rs must be able to collect large data set from the network with high frequency and resolution with minimal impact to the device's CPU and memory.

L-Data-REQ-02 (IC): I2RS must be able to use a database model where the data on the network node must be able to be described in the I2RS exchange as the data plus the structure of the data. The I2RS management system consumes and understand the data only after it consumes and understand the database model or has been trained by vendor published model

L-Data-REQ-03 (IC): I2RS should use a pub-sub model which allows scaling plus push or pull of data.

L-Data-REQ-04 (IC): I2RS should support capability negotiation to inform a subscriber of the options for publication of data. The options include transport, security, and error handling.

L-Data-REQ-05 (IC): The I2RS data transfer should be format agnostic. This means the publisher and subscriber may agree upon XML, JSON, MTL, protobufs or any other format.

L-Data-REQ-06 (IC): I2RS Transports must be able to be chosen by a I2RS Client-I2RS Agent pair. An I2RS Client-I2RS Agent pair

should be allowed to negotiate the transport options from a list of options.

L-DATA-REQ-07 (IC): The I2RS interface (protocol and IMs) should allow a subscribe to select portions of the data model.

L-Data-REQ-08 (IC): The I2RS interface (protocol and IMs) should allow for multiple publish subscriptions at a time.

L-Data-REQ-09 (IC): Timesteps should be associated with data that requires it. Not all data will require a time stamp. Additional time stamps may be added.

L-Data-REQ-10 (IC): The I2RS should support the query and "introspection" of the data model. The Introspections provides support for data verification, easier inclusion in legacy data, and easier merging with data streams.

L-Data-REQ-11 (IC): After the I2rs Client-Agent have exchanged capabilities, a database model, and filters used to select elements of the model to subscribe to, the framework should support a standard way to register for all the data desired, using whatever capabilities were advertised by the node. Once registration is complete, the control channel can be closed. Ensuring subscriptions are correct, complete, and replicated or not, is up to the overall system and not the agent on the network node.

L-Data-REQ-12 (IC): The I2RS interface should support user subscriptions to data with the following parameters:

- \* push of data synchronously or asynchronously via registered subscriptions
- \* pull data off in a one-shot pull or in multiple sequences
- \* provide dynamic subscriptions that can be setup via IPFIX feed
- \* support of subscriber and consumer I2RS Client-agent pairs
- \* allow remapping of a node's databases

L-Data-REQ-13 (IC): The I2RS interface must handle and report errors that occur with data subscription, stale data, repeated transport failures, and other (yet unknown) errors

## 14. CDNI

The requirements from the Content Delivery Network Interaction described in [I-D.shin-i2rs-usecases-cdni-request-routing] are (OC):

- o CDNI-REQ-01 (OC): The I2RS interface should support two CDNI functionalities [I-D.ietf-cdni-framework]:
  - \* Request Routing Interface - Footprint and Capabilities Advertisement; the asynchronous advertisement of footprint and capabilities by a dCDN that allows a uCDN to decide whether to redirect particular user requests to that dCDN via the ALTO protocol; and
  - \* Request Routing Interface - Redirection; the synchronous operation of actually redirecting a user request via I2RS manipulation of the routing plane.
- o CDNI-REQ-02 (OC): The I2RS (Protocol and IM) should provide facilities to enable the query/response of information from an ALTO services in a node routing functions so that the upstream CDN provider can select a proper downstream CDN provider for a given end user request.
- o CDNI-REQ-03 (OC): I2RS (protocol and IM) should provide facilities to enable I2RS can help the upstream CDN provider to redirect a content request message to a downstream CDN provider for a given end user request as with the following features:
  - \* The uCDN relays this message between I2RS Clients and I2RS agents with content distribution metadata, and queries the dCDN whether user request message can be delivered. This query can have multiple dCDN that the user message can be delivered to.
  - \* the I2RS agent associated with the dCDN delivery requests indicating which dCDN (if any) the user message can be delivered to.
  - \* Allow dCDN to be managed to deliver content by having the messages to signal back to the uCDN the (destination (?)) iP address for the content, on the dCDN, and the pathway between the uCDN for surrogate deliver via the dCDN of user data. Part of this management is the passing of URL of the surrogate in dCDN (for HTTP Redirection to be transmitting) back from the dCDN to the uCDN so the uCDN can inform the end user.

## 15. IANA Considerations

This document makes no request of IANA.

## 16. Security Considerations

Routing information is very critical and sensitive information for the operators. I2RS should provide strong security mechanism to protect the routing information that it could not be accessed by the un-authorized users. It should also protect the security and integrity protection of the routing data.

## 17. References

## 17.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3746] Yang, L., Dantu, R., Anderson, T., and R. Gopal, "Forwarding and Control Element Separation (ForCES) Framework", RFC 3746, April 2004.

## 17.2. Informative References

- [I-D.amante-i2rs-topology-use-cases]  
Medved, J., Previdi, S., Lopez, V., and S. Amante,  
"Topology API Use Cases", draft-amante-i2rs-topology-use-cases-01 (work in progress), October 2013.
- [I-D.bitar-i2rs-service-chaining]  
Bitar, N., Heron, G., Fang, L., ramki, r., Leymann, N.,  
Shah, H., and W. Haddad, "Interface to the Routing System  
(I2RS) for Service Chaining: Use Cases and Requirements",  
draft-bitar-i2rs-service-chaining-01 (work in progress),  
February 2014.
- [I-D.chen-i2rs-mpls-ldp-usecases]  
Chen, X. and Z. Li, "Use Cases for an Interface to LDP  
Protocol", draft-chen-i2rs-mpls-ldp-usecases-00 (work in  
progress), October 2013.
- [I-D.chen-i2rs-ts-use-case]  
Chen, M. and S. Hares, "I2RS Traffic Steering Use Case",  
draft-chen-i2rs-ts-use-case-01 (work in progress), July  
2014.



- [I-D.hares-i2rs-use-case-vn-vc]  
Hares, S. and M. Chen, "Use Cases for Virtual Connections on Demand (VCoD) and Virtual Network on Demand (VNoD) using Interface to Routing System", draft-hares-i2rs-use-case-vn-vc-03 (work in progress), July 2014.
- [I-D.huang-i2rs-mpls-te-usecases]  
Huang, T., Li, Z., and S. Hares, "Use Cases for an Interface to MPLS TE", draft-huang-i2rs-mpls-te-usecases-02 (work in progress), July 2014.
- [I-D.ietf-i2rs-architecture]  
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-09 (work in progress), March 2015.
- [I-D.ietf-i2rs-problem-statement]  
Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Problem Statement", draft-ietf-i2rs-problem-statement-06 (work in progress), January 2015.
- [I-D.ietf-i2rs-rib-info-model]  
Bahadur, N., Folkes, R., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-06 (work in progress), March 2015.
- [I-D.ietf-sfc-problem-statement]  
Quinn, P. and T. Nadeau, "Service Function Chaining Problem Statement", draft-ietf-sfc-problem-statement-13 (work in progress), February 2015.
- [I-D.ji-i2rs-usecases-ccne-service]  
Ji, X., Zhuang, S., Huang, T., and S. Hares, "I2RS Use Cases for Control of Forwarding Path by Central Control Network Element (CCNE)", draft-ji-i2rs-usecases-ccne-service-02 (work in progress), July 2014.
- [I-D.keyupate-i2rs-bgp-usecases]  
Patel, K., Fernando, R., Gredler, H., Amante, S., White, R., and S. Hares, "Use Cases for an Interface to BGP Protocol", draft-keyupate-i2rs-bgp-usecases-04 (work in progress), July 2014.

- [I-D.krishnan-i2rs-large-flow-use-case]  
ramki, r., Ghanwani, A., Kini, S., McDysan, D., and D. Lopez, "Large Flow Use Cases for I2RS PBR and QoS", draft-krishnan-i2rs-large-flow-use-case-04 (work in progress), April 2014.
- [I-D.lapukhov-bgp-routing-large-dc]  
Lapukhov, P., Premji, A., and J. Mitchell, "Use of BGP for routing in large-scale data centers", draft-lapukhov-bgp-routing-large-dc-06 (work in progress), August 2013.
- [I-D.medved-i2rs-topology-requirements]  
Medved, J., Previdi, S., Gredler, H., Nadeau, T., and S. Amante, "Topology API Requirements", draft-medved-i2rs-topology-requirements-00 (work in progress), February 2013.
- [I-D.shin-i2rs-usecases-cdni-request-routing]  
Shin, M. and S. Lee, "CDNI Request Routing with I2RS", draft-shin-i2rs-usecases-cdni-request-routing-00 (work in progress), July 2014.
- [I-D.swhyte-i2rs-data-collection-system]  
Whyte, S., Hines, M., and W. Kumari, "Bulk Network Data Collection System", draft-swhyte-i2rs-data-collection-system-00 (work in progress), October 2013.
- [I-D.white-i2rs-use-case]  
White, R., Hares, S., and A. Retana, "Protocol Independent Use Cases for an Interface to the Routing System", draft-white-i2rs-use-case-06 (work in progress), July 2014.
- [I-D.zhang-i2rs-mbb-usecases]  
Zhang, L., Li, Z., Liu, D., and S. Hares, "Use Cases of I2RS in Mobile Backhaul Network", draft-zhang-i2rs-mbb-usecases-01 (work in progress), February 2014.
- [RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, August 2006.
- [RFC5212] Shiimoto, K., Papadimitriou, D., Le Roux, J.L., Vigoureux, M., and D. Brungard, "Requirements for GMPLS-Based Multi-Region and Multi-Layer Networks (MRN/MLN)", RFC 5212, July 2008.
- [RFC5286] Atlas, A. and A. Zinin, "Basic Specification for IP Fast Reroute: Loop-Free Alternates", RFC 5286, September 2008.

- [RFC5623] Oki, E., Takeda, T., Le Roux, JL., and A. Farrel,  
"Framework for PCE-Based Inter-Layer MPLS and GMPLS  
Traffic Engineering", RFC 5623, September 2009.
- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic  
Optimization (ALTO) Problem Statement", RFC 5693, October  
2009.

Authors' Addresses

Susan Hares  
Huawei

Email: shares@ndzh.com

Mach Chen  
Huawei Technologies  
Huawei Bld., No.156 Beiqing Rd.  
Beijing 100095  
China

Email: mach.chen@huawei.com

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: October 24, 2016

A. Atlas  
Juniper Networks  
J. Halpern  
Ericsson  
S. Hares  
Huawei  
D. Ward  
Cisco Systems  
T. Nadeau  
Brocade  
April 22, 2016

An Architecture for the Interface to the Routing System  
draft-ietf-i2rs-architecture-15

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on October 24, 2016.

Copyright Notice

Copyright (c) 2016 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. Drivers for the I2RS Architecture . . . . .	4
1.2. Architectural Overview . . . . .	5
2. Terminology . . . . .	9
3. Key Architectural Properties . . . . .	12
3.1. Simplicity . . . . .	12
3.2. Extensibility . . . . .	12
3.3. Model-Driven Programmatic Interfaces . . . . .	13
4. Security Considerations . . . . .	14
4.1. Identity and Authentication . . . . .	16
4.2. Authorization . . . . .	16
4.3. Client Redundancy . . . . .	17
4.4. I2RS in Personal Devices . . . . .	17
5. Network Applications and I2RS Client . . . . .	18
5.1. Example Network Application: Topology Manager . . . . .	18
6. I2RS Agent Role and Functionality . . . . .	19
6.1. Relationship to its Routing Element . . . . .	19
6.2. I2RS State Storage . . . . .	19
6.2.1. I2RS Agent Failure . . . . .	20
6.2.2. Starting and Ending . . . . .	21
6.2.3. Reversion . . . . .	21
6.3. Interactions with Local Configuration . . . . .	21
6.3.1. Examples of Local Configuration vs. I2RS Ephemeral Configuration . . . . .	23
6.4. Routing Components and Associated I2RS Services . . . . .	24
6.4.1. Routing and Label Information Bases . . . . .	25
6.4.2. IGPs, BGP and Multicast Protocols . . . . .	26
6.4.3. MPLS . . . . .	26
6.4.4. Policy and QoS Mechanisms . . . . .	27
6.4.5. Information Modeling, Device Variation, and Information Relationships . . . . .	27
6.4.5.1. Managing Variation: Object Classes/Types and Inheritance . . . . .	27
6.4.5.2. Managing Variation: Optionality . . . . .	28
6.4.5.3. Managing Variation: Templating . . . . .	28
6.4.5.4. Object Relationships . . . . .	29

6.4.5.4.1.	Initialization . . . . .	29
6.4.5.4.2.	Correlation Identification . . . . .	29
6.4.5.4.3.	Object References . . . . .	30
6.4.5.4.4.	Active Reference . . . . .	30
7.	I2RS Client Agent Interface . . . . .	30
7.1.	One Control and Data Exchange Protocol . . . . .	30
7.2.	Communication Channels . . . . .	31
7.3.	Capability Negotiation . . . . .	31
7.4.	Scope Policy Specifications . . . . .	32
7.5.	Connectivity . . . . .	32
7.6.	Notifications . . . . .	33
7.7.	Information collection . . . . .	33
7.8.	Multi-Headed Control . . . . .	34
7.9.	Transactions . . . . .	34
8.	Operational and Manageability Considerations . . . . .	35
9.	IANA Considerations . . . . .	36
10.	Acknowledgements . . . . .	36
11.	References . . . . .	36
11.1.	Normative References . . . . .	36
11.2.	Informative References . . . . .	36
	Authors' Addresses . . . . .	38

## 1. Introduction

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

Routers convert all of this information into forwarding entries which are then used to forward packets and flows between network elements. The forwarding plane and the specified forwarding entries then contain active state information that describes the expected and observed operational behavior of the router and which is also needed by the network applications. Network-oriented applications require easy access to this information to learn the network topology, to verify that programmed state is installed in the forwarding plane, to measure the behavior of various flows, routes or forwarding entries, as well as to understand the configured and active states of the router. Network-oriented applications also require easy access to an interface which will allow them to program and control state related to forwarding.

This document sets out an architecture for a common, standards-based interface to this information. This Interface to the Routing System (I2RS) facilitates control and observation of the routing-related

state (for example, a Routing Element RIB manager's state), as well as enabling network-oriented applications to be built on top of today's routed networks. The I2RS is a programmatic asynchronous interface for transferring state into and out of the internet routing system. This I2RS architecture recognizes that the routing system and a router's Operating System (OS) provide useful mechanisms that applications could harness to accomplish application-level goals. These network-oriented applications can leverage the I2RS programmatic interface to create new ways of combining retrieval of internet routing data, analyzing this data, setting state within routers.

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 way for applications to customize network behavior while leveraging the existing routing system as desired. The I2RS provides a framework for applications (including controller applications) to register and to request the appropriate information for each particular application.

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

Security is a concern for any new interface to the routing system. Section 4 provides an overview of the security considerations for the I2RS architecture. The detailed requirements for I2RS protocol security are contained in [I-D.ietf-i2rs-protocol-security-requirements], and the detailed security requirements for environment in which the I2RS protocol exists in are contained in [I-D.ietf-i2rs-security-environment-reqs].

### 1.1. Drivers for the I2RS Architecture

There are four key drivers that shape the I2RS architecture. First is the need for an interface that is programmatic, asynchronous, and offers fast, interactive access for atomic operations. Second is the access to structured information and state that is frequently not directly configurable or modeled in existing implementations or configuration protocols. Third is the ability to subscribe to structured, filterable event notifications from the router. Fourth, the operation of I2RS is to be data-model driven to facilitate extensibility and provide standard data-models to be used by network applications.

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

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

Such an interface also facilitates the injection of ephemeral state into the routing system. Ephemeral state on a router is the state which does not survive a the reboot of a routing device or the reboot of the software handling the I2RS software on a routing device. A non-routing protocol or application could inject state into a routing element via the state-insertion functionality of the I2RS and that state could then be distributed in a routing or signaling protocol and/or be used locally (e.g. to program the co-located forwarding plane). I2RS will only permit modification of state that would be possible to modify via local configuration; no direct manipulation of protocol-internal dynamically determined data is envisioned.

## 1.2. Architectural Overview

Figure 1 shows the basic architecture for I2RS between applications using I2RS, their associated I2RS clients, and I2RS agents. Applications access I2RS services through I2RS clients. A single I2RS client can provide access to one or more applications. This figure also shows the types of data models associated with the routing system (dynamic configuration, static configuration, local configuration, and routing and signaling configuration) which the I2RS agent data models may access or augment.

Figure 1 is similar to the figure 1 found in the [I-D.ietf-i2rs-problem-statement], but this figure shows additional detail on how the applications utilize I2RS clients to interact with I2RS agents. Figure 1 also shows a logical view of the data models associated with the routing system rather than a functional view (RIB, FIB, topology, policy, routing/signaling protocols, etc.)

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

Applications can access I2RS services through local or remote clients. A local client operates on the same physical box as routing system. In contrast, a remote client operates across the network. In the figure, Applications A and B access I2RS services through local clients, while Applications C, D and E access I2RS services



through a remote client. The details of how applications communicate with a remote client is out of scope for I2RS.

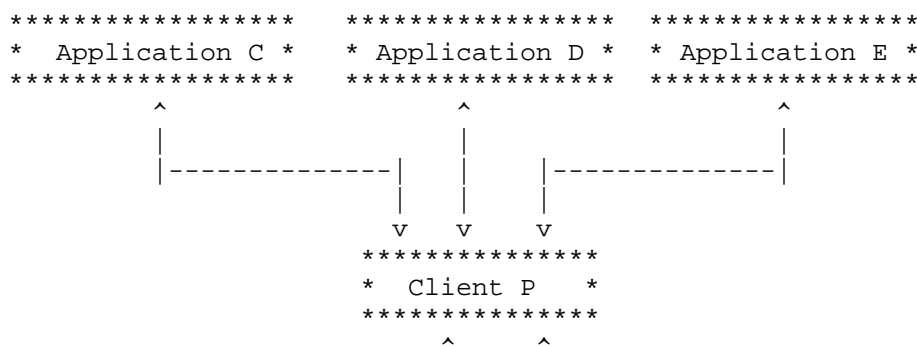
An I2RS client can access one or more I2RS agents. In the figure 1, Clients B and P access I2RS agents 1 and 2. Likewise, an I2RS agent can provide service to one or more clients. In this figure, I2RS agent 1 provides services to Clients A, B and P while Agent 2 provides services to only Clients B and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

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

The I2RS agent provides read and write access to selected data on the routing element that are organized into I2RS Services. Section 4 describes how access is mediated by authentication and access control mechanisms. Figure 1 shows I2RS agents being able to write ephemeral static state (e.g. RIB entries), and to read from dynamic static (e.g. MPLS LSP-ID or number of active BGP peers).

In addition to read and write access, the I2RS agent allows clients to subscribe to different types of notifications about events affecting different object instances. One example of a notification of such an event (which is unrelated to an object creation, modification or deletion) is when a next-hop in the RIB is resolved in a way that allows it to be used by a RIB manager for installation in the forwarding plane as part of a particular route. Please see Section 7.6 and Section 7.7 for details.

The scope of I2RS is to define the interactions between the I2RS agent and the I2RS client and the associated proper behavior of the I2RS agent and I2RS client.



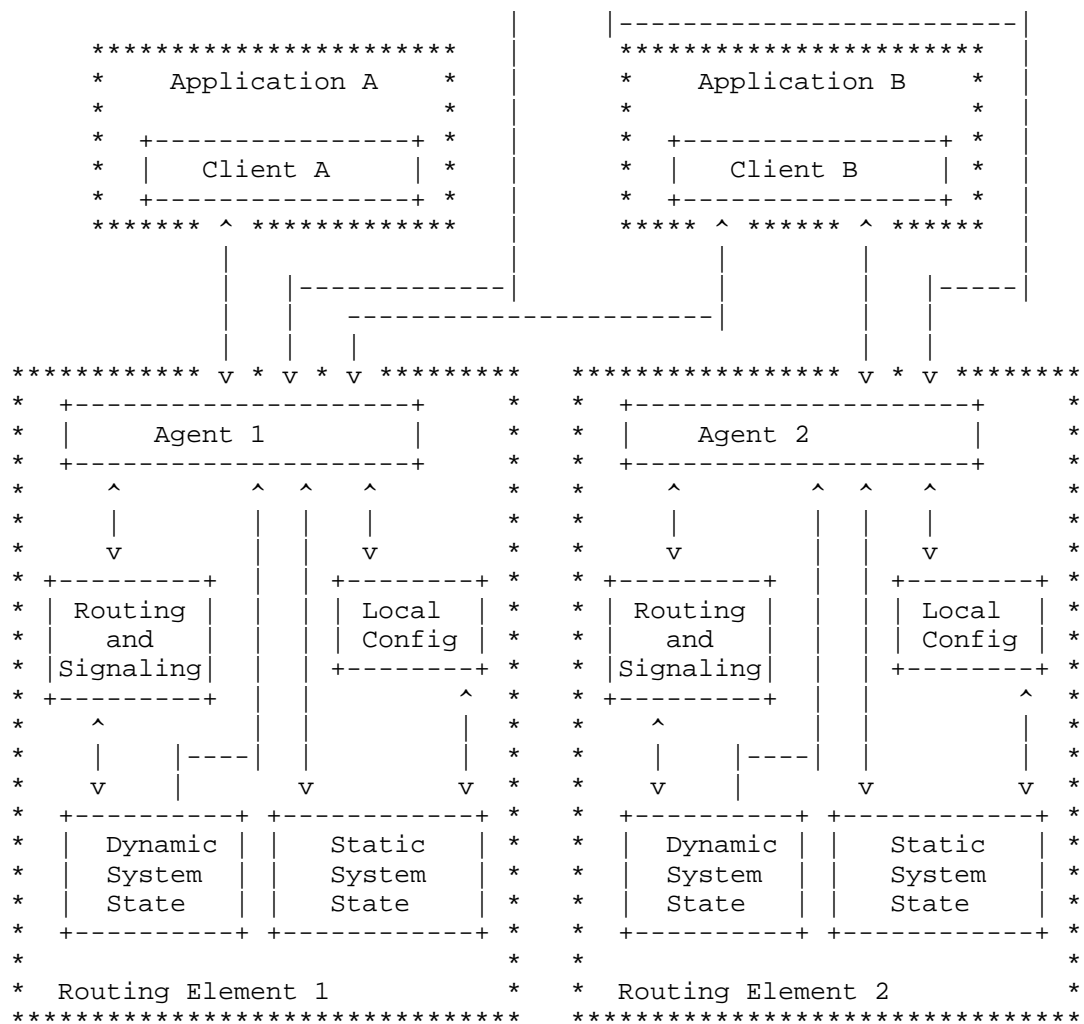


Figure 1: Architecture of I2RS Clients and Agents

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

- \* A router with a forwarding plane and RIB Manager that runs IS-IS, OSPF, BGP, PIM, etc.,
- \* A BGP speaker acting as a Route Reflector,

- \* An LSR that implements RSVP-TE, OSPF-TE, and PCEP and has a forwarding plane and associated RIB Manager,
- \* A server that runs IS-IS, OSPF, BGP and uses ForCES to control a remote forwarding plane,

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

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

**Local Configuration:** is the black box behavior for interactions between the ephemeral state that I2RS installs into the routing element; and this Local Configuration is defined by this document and the behaviors specified by the I2RS protocol.

**Dynamic System State:** An I2RS agent needs access to state on a routing element beyond what is contained in the routing subsystem. Such state may include various counters, statistics, flow data, and local events. This is the subset of operational state that is needed by network applications based on I2RS that is not contained in the routing and signaling information. How this information is provided to the I2RS agent is out of scope, but the standardized information and data models for what is exposed are part of I2RS.

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

**I2RS agent:** See the definition in Section 2.

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

**I2RS client:** See the definition in Section 2.

As can be seen in Figure 1, an I2RS client can communicate with multiple I2RS agents. Similarly, an I2RS agent may communicate with multiple I2RS clients - whether to respond to their requests, to send notifications, etc. Timely notifications are critical so that

several simultaneously operating applications have up-to-date information on the state of the network.

As can also be seen in Figure 1, an I2RS agent may communicate with multiple clients. Each client may send the agent a variety of write operations. In order to keep the protocol simple, two clients should not attempt to write (modify) the same piece of information on an I2RS agent. This is considered an error. However, such collisions may happen and section 7.8 (multi-headed control) describes how the I2RS agent resolves collision by first utilizing priority to resolve collisions, and second by servicing the requests in a first in, first served basis. The I2RS architecture includes this definition of behavior for this case simply for predictability not because this is an intended result. This predictability will simplify the error handling and suppress oscillations. If additional error cases beyond this simple treatment are required, these error cases should be resolved by the network applications and management systems.

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

The architectural goal for the I2RS is that such errors should produce predictable behaviors, and be reportable to interested clients. The details of the associated policy is discussed in Section 7.8. The same policy mechanism (simple priority per I2RS client) applies to interactions between the I2RS agent and the CLI/SNMP/NETCONF as described in Section 6.3.

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

## 2. Terminology

The following terminology is used in this document.

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

**client or I2RS client:** A client implements the I2RS protocol, uses it to communicate with I2RS agents, and uses the I2RS services to accomplish a task. It interacts with other elements of the policy, provisioning, and configuration system by means outside of the scope of the I2RS effort. It interacts with the I2RS agents to collect information from the routing and forwarding system. Based on the information and the policy oriented interactions, the I2RS client may also interact with I2RS agents to modify the state of their associated routing systems to achieve operational goals. An I2RS client can be seen as the part of an application that uses and supports I2RS and could be a software library.

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

**read scope:** The read scope of an I2RS client within an I2RS agent is the set of information which the I2RS client is authorized to read within the I2RS agent. The read scope specifies the access restrictions to both see the existence of data and read the value of that data.

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

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

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

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

role or security role: A security role specifies the scope, resources, priorities, etc. that a client or agent has. If a identity has multiple roles in the security system, the identity is permitted to perform any operations any of those roles permit. Multiple identities may use the same security role.

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

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

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

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

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

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

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

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

routing system/subsystem: is a set of software and hardware that handles determining where packets are forwarded to which the I2RS

system connects. The term "packets" may be qualified to be layer 1 frames, layer 2 frames or layer 3 packets. The phrase "Internet routing system" implies the packets which have IP as layer 3. A routing "subsystem" indicates that the routing software/hardware is only the subsystem of another larger system.

### 3. Key Architectural Properties

Several key architectural properties for the I2RS protocol are elucidated below (simplicity, extensibility, and model-driven programmatic interfaces). However, some architecture properties such as performance and scaling are not described below because they are discussed in [I-D.ietf-i2rs-problem-statement], may vary based on the particular use-cases.

#### 3.1. Simplicity

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

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

Thus, one of the key aims for I2RS is to keep the protocol and modeling architecture simple. So for each architectural component or aspect, we ask ourselves "do we need this complexity, or is the behavior merely nice to have?" If we need the complexity, we should ask ourselves "Is this the simplest way to provide in the I2RS external interface?"

#### 3.2. Extensibility

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

The scope of I2RS work is being designed in phases to provide deliverable and deployable results at every phase. Each phase will have a specific set of requirements, and the I2RS protocol and data models will progress toward these requirements. Therefore, it is clearly desirable for the I2RS data models to be easily and highly extensible to represent additional aspects of the network elements or network systems. It should be easy to integrate data models from the I2RS with other data. This reinforces the criticality of designing the data models to be highly extensible, preferably in a regular and simple fashion.

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

### 3.3. Model-Driven Programmatic Interfaces

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

There are several key benefits for I2RS in using model-driven architecture and protocol(s). First, it allows for data-model focused processing of management data that provides modular implementation in I2RS clients and I2RS agents. The I2RS client only needs to implement the models the I2RS client is able to access. The I2RS agent only needs to implement the data models the I2RS agent supports.

Second, tools can automate checking and manipulating data; this is particularly valuable for both extensibility and for the ability to easily manipulate and check proprietary data-models.

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

The purpose of the data model is to provide a definition of the information regarding the routing system that can be used in operational networks. If routing information is being modeled for



the first time, a logical information model may be standardized prior to creating the data model.

#### 4. Security Considerations

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

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

Due to the re-use strategy of the I2RS architecture, this security section describes the assumed security environment for I2RS with additional details on: a) identity and authentication, b) authorization, and c) client redundancy. Each protocol proposed for inclusion as an I2RS protocol will need to be evaluated for the security constraints of the protocol. The detailed requirements for the I2RS protocol and the I2RS security environment will be defined within these global security environments.

The I2RS protocol security requirements for I2RS protocol version 1 are contained in [I-D.ietf-i2rs-protocol-security-requirements], and the global I2RS security environment requirements for protocol version 1 are contained [I-D.ietf-i2rs-security-environment-reqs].

First, here is a brief description of the assumed security environment for I2RS. The I2RS agent associated with a Routing Element is a trusted part of that Routing Element. For example, it may be part of a vendor-distributed signed software image for the entire Routing Element or it may be trusted signed application that an operator has installed. The I2RS agent is assumed to have a separate authentication and authorization channel by which it can validate both the identity and permissions associated with an I2RS client. To support numerous and speedy interactions between the I2RS agent and I2RS client, it is assumed that the I2RS agent can also cache that particular I2RS clients are trusted and their associated authorized scope. This implies that the permission information may

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

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

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

If the I2RS client is acting as a broker for multiple applications, then managing the security, authentication and authorization for that communication is out of scope; nothing prevents the broker from using I2RS protocol and a separate authentication and authorization channel from being used. Regardless of mechanism, an I2RS client that is acting as a broker is responsible for determining that applications using it are trusted and permitted to make the particular requests.

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

Other communications via I2RS may not require integrity, confidentiality, and replay protection. For instance, if an I2RS client subscribes to an information stream of prefix announcements from OSPF, those may require integrity but probably not confidentiality or replay protection. Similarly, an information stream of interface statistics may not even require guaranteed delivery. In Section 7.2, additional login regarding multiple communication channels and their use is provided. From the security perspective, it is critical to realize that an I2RS agent may open a new communication channel based upon information provided by an I2RS client (as described in Section 7.2). For example, an I2RS client may request notifications of certain events and the agent will open a

communication channel to report such events. Therefore, to avoid an indirect attack, such a request must be done in the context of an authenticated and authorized client whose communications cannot have been altered.

#### 4.1. Identity and Authentication

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

I2RS clients may be operating on behalf of other applications. While those applications' identities are not needed for authentication or authorization, each application should have a unique opaque identifier that can be provided by the I2RS client to the I2RS agent for purposes of tracking attribution of operations to an application identifier (and from that to the application's identity). This tracking of operations to an application supports I2RS functionality for tracing actions (to aid troubleshooting in routers) and logging of network changes.

#### 4.2. Authorization

All operations using I2RS, both observation and manipulation, should be subject to appropriate authorization controls. Such authorization is based on the identity and assigned role of the I2RS client performing the operations and the I2RS agent in the network element. Multiple Identities may use the same role(s). As noted in the definition of the identity and role above, if multiple roles are associated with an identity then the identity is authorized to perform any operation authorized by any of its roles.

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

The appropriate or necessary level of granularity for scope can depend upon the particular I2RS Service and the implementation's granularity. An approach to a similar access control problem is defined in the NetConf Access Control Model (NACM) [RFC6536]; it allows arbitrary access to be specified for a data node instance

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

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

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

#### 4.3. Client Redundancy

I2RS must support client redundancy. At the simplest, this can be handled by having a primary and a backup network application that both use the same client identity and can successfully authenticate as such. Since I2RS does not require a continuous transport connection and supports multiple transport sessions, this can provide some basic redundancy. However, it does not address the need for troubleshooting and logging of network changes to be informed about which network application is actually active. At a minimum, basic transport information about each connection and time can be logged with the identity.

#### 4.4. I2RS in Personal Devices

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

## 5. Network Applications and I2RS Client

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

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

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

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

### 5.1. Example Network Application: Topology Manager

A Topology Manager includes an I2RS client that uses the I2RS data models and protocol to collect information about the state of the network by communicating directly with one or more I2RS agents. From these I2RS agents, the Topology Manager collects routing configuration and operational data, such as interface and label-switched path (LSP) information. In addition, the Topology Manager may collect link-state data in several ways - either via I2RS models, by peering with BGP-LS[RFC7752] or listening into the IGP.

The set of functionality and collected information that is the Topology Manager may be embedded as a component of a larger application, such as a path computation application. As a stand-alone application, the Topology Manager could be useful to other

network applications by providing a coherent picture of the network state accessible via another interface. That interface might use the same I2RS protocol and could provide a topology service using extensions to the I2RS data models.

## 6. I2RS Agent Role and Functionality

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

### 6.1. Relationship to its Routing Element

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

For scalability and generality, the I2RS agent may be responsible for collecting and delivering large amounts of data from various parts of the routing element. Those parts may or may not actually be part of a single physical device. Thus, for scalability and robustness, it is important that the architecture allow for a distributed set of reporting components providing collected data from the I2RS agent back to the relevant I2RS clients. There may be multiple I2RS agents within the same router. In such a case, they must have non-overlapping sets of information which they manipulate.

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

### 6.2. I2RS State Storage

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

### 6.2.1. I2RS Agent Failure

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

**NOTIFICATION\_I2RS\_AGENT\_TERMINATING:** This notification reports that the associated I2RS agent is shutting down gracefully, and that I2RS ephemeral state will be removed. It can optionally include a timestamp indicating when the I2RS agent will shutdown. Use of this timestamp assumes that time synchronization has been done and the timestamp should not have granularity finer than one second because better accuracy of shutdown time is not guaranteed.

**NOTIFICATION\_I2RS\_AGENT\_STARTING:** This notification signals to the I2RS client(s) that the associated I2RS agent has started. It includes an agent-boot-count that indicates how many times the I2RS agent has restarted since the associated routing element restarted. The agent-boot-count allows an I2RS client to determine if the I2RS agent has restarted. (Note: This notification will on be sent by the I2RS agent to I2RS clients which are known by the I2RS agent after a reboot. How the I2RS agent retains the knowledge of these I2RS clients is out of scope of this architecture.)

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

**Unexpected failure:** In this case, the I2RS agent has unexpectedly crashed and thus cannot notify its clients of anything. Since I2RS does not require a persistent connection between the I2RS client and I2RS agent, it is necessary to have a mechanism for the I2RS agent to notify I2RS clients that had subscriptions or written ephemeral state; such I2RS clients should be cached by the I2RS agent's system in persistent storage. When the I2RS agent starts, it should send a NOTIFICATION\_I2RS\_AGENT\_STARTING to each cached I2RS client.

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

#### 6.2.2. Starting and Ending

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

#### 6.2.3. Reversion

An I2RS agent may decide that some state should no longer be applied. An I2RS client may instruct an agent to remove state it has applied. In all such cases, the state will revert to what it would have been without the I2RS client-agent interaction; that state is generally whatever was specified via the CLI, `NETCONF`, `SNMP`, etc. I2RS agents will not store multiple alternative states, nor try to determine which one among such a plurality it should fall back to. Thus, the model followed is not like the RIB, where multiple routes are stored at different preferences. (For I2RS state in the presence of two I2RS clients, please see section 1.2 and section 7.8)

An I2RS client may register for notifications, subject to its notification scope, regarding state modification or removal by a particular I2RS client.

#### 6.3. Interactions with Local Configuration

Changes may originate from either Local Configuration or from I2RS. The modifications and data stored by I2RS are separate from the local device configuration, but conflicts between the two must be resolved in a deterministic manner that respects operator-applied policy. The deterministic manner is the result of general I2RS rules, system rules, knobs adjusted by operator-applied policy, and the rules associated with the YANG data model (often in `MUST` and `WHEN` clauses for dependencies).

The operator-applied policy knobs can determine whether the Local Configuration overrides a particular I2RS client's request or vice versa. Normally, most devices will have an operator-applied policy



that will prioritize the I2RS client's ephemeral configuration changes so that ephemeral data overrides the Local Configuration.

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

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

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

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

When the Local Configuration always wins, some communication between that subsystem and the I2RS agent is still necessary. As an I2RS agent connects to the routing sub-system, the I2RS agent must also communicate with the Local Configuration to exchange model information so the I2RS agent knows the details of each specific device configuration change that the I2RS agent is permitted to modify. In addition, when the system determines, that a client's I2RS state is preempted, the I2RS agent must notify the affected I2RS clients; how the system determines this is implementation-dependent.

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

### 6.3.1. Examples of Local Configuration vs. I2RS Ephemeral Configuration

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

Policy Knob 1: Ephemeral configuration overwrites local configuration.

Policy Knob 2: Update of local configuration value supercedes and overwrites the ephemeral configuration.

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

1. Does the operator policy allow the ephemeral configuration changes to have priority over existing local configuration?
2. Does the YANG data model have any rules associated with the ephemeral configuration (such as "MUST" or "WHEN" rule)?

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

The policy settings are:

	Policy Knob 1 =====	Policy Knob 2 =====
Router A	ephemeral has priority	ephemeral has priority
Router B	local config has priority	local config has priority
Router C	ephemeral has priority	local config has priority

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

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

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

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

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

For simplicity, each logical protocol or set of functionality that can be compactly described in a separable information and data model is considered as a separate I2RS Service. A routing element need not implement all routing components described nor provide the associated I2RS services. I2RS Services should include a capability model so that peers can determine which parts of the service are supported. Each I2RS Service requires an information model that describes at least the following: data that can be read, data that can be written, notifications that can be subscribed to, and the capability model mentioned above.

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

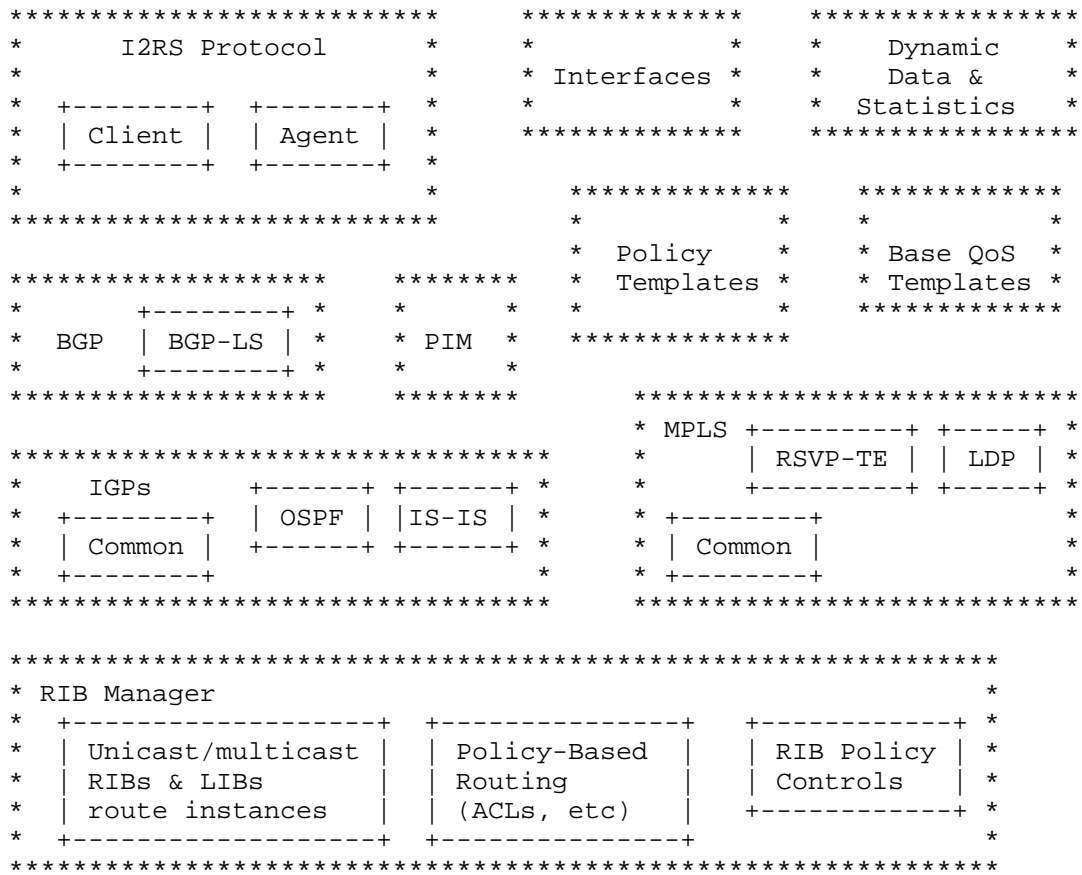


Figure 2: Anticipated I2RS Services

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

#### 6.4.1. Routing and Label Information Bases

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

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

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

#### 6.4.2. IGP, BGP and Multicast Protocols

A separate I2RS Service can expose each routing protocol on the device. Such I2RS services may include a number of different kinds of operations:

- o reading the various internal RIB(s) of the routing protocol is often helpful for understanding the state of the network. Directly writing to these protocol-specific RIBs or databases is out of scope for I2RS.
- o reading the various pieces of policy information the particular protocol instance is using to drive its operations.
- o writing policy information such as interface attributes that are specific to the routing protocol or BGP policy that may indirectly manipulate attributes of routes carried in BGP.
- o writing routes or prefixes to be advertised via the protocol.
- o joining/removing interfaces from the multicast trees.
- o subscribing to an information stream of route changes
- o receiving notifications about peers coming up or going down.

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

#### 6.4.3. MPLS

I2RS Services will be needed to expose the protocols that create transport LSPs (e.g. LDP and RSVP-TE) as well as protocols (e.g. BGP, LDP) that provide MPLS-based services (e.g. pseudowires, L3VPNs,

L2VPNs, etc). This should include all local information about LSPs originating in, transiting, or terminating in this Routing Element.

#### 6.4.4. Policy and QoS Mechanisms

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

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

#### 6.4.5. Information Modeling, Device Variation, and Information Relationships

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

##### 6.4.5.1. Managing Variation: Object Classes/Types and Inheritance

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

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

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

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

#### 6.4.5.2. Managing Variation: Optionality

I2RS Information Models must be clear about what aspects are optional. For instance, must an instance of a class always contain a particular data field X? If so, must the client provide a value for X when creating the object or is there a well-defined default value? From the Routing Element perspective, in the above example, each Information model should provide information that:

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

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

#### 6.4.5.3. Managing Variation: Templating

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

For instance, assigning traffic to a particular service class might be done by specifying a template Queueing with a parameter to indicate Gold, Silver, or Best Effort. The details of how that is carried out are not modeled. This does assume that the necessary templates are made available on the Routing Element via some

mechanism other than I2RS. The idea is that by providing suitable templates for tasks that need to be accomplished, with templates implemented differently for different kinds of Routing Elements, the client can easily interact with the Routing Element without concern for the variations which are handled by values included in the template.

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

#### 6.4.5.4. Object Relationships

Objects (in a Routing Element or otherwise) do not exist in isolation. They are related to each other. One of the important things a class definition does is represent the relationships between instances of different classes. These relationships can be very simple, or quite complicated. The following lists the information relationships that the information models need to support.

##### 6.4.5.4.1. Initialization

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

##### 6.4.5.4.2. Correlation Identification

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



#### 6.4.5.4.3. Object References

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

#### 6.4.5.4.4. Active Reference

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

### 7. I2RS Client Agent Interface

#### 7.1. One Control and Data Exchange Protocol

This I2RS architecture assumes a data-model driven protocol where the data-models are defined in YANG 1.1 ([I-D.ietf-netmod-rfc6020bis]), and associated YANG based model drafts ([RFC6991], [RFC7223], [RFC7224], [RFC7277], [RFC7317]). Two the protocols to be expanded to support the I2RS protocol are NETCONF [RFC6241] and RESTCONF [I-D.ietf-netconf-restconf]. This helps meet the goal of simplicity and thereby enhances deployability. The I2RS protocol may need to use several underlying transports (TCP, SCTP (stream control transport protocol), DCCP (Datagram Congestion Control Protocol)), with suitable authentication and integrity protection mechanisms. These different transports can support different types of communication (e.g. control, reading, notifications, and information collection) and different sets of data. Whatever transport is used for the data exchange, it must also support suitable congestion control mechanisms. The transports chosen should be operator and implementor friendly to ease adoption.

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

## 7.2. Communication Channels

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

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

## 7.3. Capability Negotiation

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

Capability negotiation (such as which transports are supported beyond the minimum required to implement) will clearly be necessary. It is important that such negotiations be kept simple and robust, as such mechanisms are often a source of difficulty in implementation and deployment.

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

#### 7.4. Scope Policy Specifications

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

#### 7.5. Connectivity

An I2RS client may or may not maintain an active communication channel with an I2RS agent. Therefore, an I2RS agent may need to open a communication channel to the client to communicate previously requested information. The lack of an active communication channel does not imply that the associated I2RS client is non-functional. When communication is required, the I2RS agent or I2RS client can open a new communication channel.

State held by an I2RS agent that is owned by an I2RS client should not be removed or cleaned up when a client is no longer communicating - even if the agent cannot successfully open a new communication channel to the client.

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

Some examples of such a mechanism include the following. In one option, the client could request state clean-up if a particular transport session is terminated. The second is to allow state expiration, expressed as a policy associated with the I2RS client's role. The state expiration could occur after there has been no successful communication channel to or from the I2RS client for the policy-specified duration.

## 7.6. Notifications

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

Such events may be of interest to multiple I2RS clients controlling data handled by an I2RS agent, and to multiple other I2RS clients which are collecting information without exerting control. The architecture therefore requires that it be practical for I2RS clients to register for a range of notifications, and for the I2RS agents to send notifications to a number of clients. The I2RS client should be able to filter the specific notifications that will be received; the specific types of events and filtering operations can vary by information model and need to be specified as part of the information model.

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

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

## 7.7. Information collection

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

### 7.8. Multi-Headed Control

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

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

In order for this approach to multi-headed control to be useful for I2RS clients, it is important that it is possible for an I2RS client to register for changes to any changes made by I2RS to data that it may care about. This is included in the I2RS event mechanisms. This also needs to apply to changes made by CLI/NETCONF/SNMP within the write-scope of the I2RS agent, as the same priority mechanism (even if it is "CLI always wins") applies there. The I2RS client may then respond to the situation as it sees fit.

### 7.9. Transactions

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

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

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

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

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

## 8. Operational and Manageability Considerations

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

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

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

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

Traceability of Interactions: The ability to trace the interactions of the requests received by the I2RS agent's and actions taken by the I2RS agents is needed so that operations can monitor I2RS

agents during deployment, and troubleshoot software or network problems.

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

## 9. IANA Considerations

This document includes no request to IANA.

## 10. Acknowledgements

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

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

## 11. References

### 11.1. Normative References

[I-D.ietf-i2rs-problem-statement]  
Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Problem Statement", draft-ietf-i2rs-problem-statement-10 (work in progress), February 2016.

### 11.2. Informative References

[I-D.ietf-i2rs-protocol-security-requirements]  
Hares, S., Migault, D., and J. Halpern, "I2RS Security Related Requirements", draft-ietf-i2rs-protocol-security-requirements-03 (work in progress), March 2016.

[I-D.ietf-i2rs-security-environment-reqs]  
Migault, D., Halpern, J., and S. Hares, "I2RS Environment Security Requirements", draft-ietf-i2rs-security-environment-reqs-01 (work in progress), April 2016.

- [I-D.ietf-netconf-restconf]  
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-12 (work in progress), April 2016.
- [I-D.ietf-netmod-rfc6020bis]  
Bjorklund, M., "The YANG 1.1 Data Modeling Language", draft-ietf-netmod-rfc6020bis-11 (work in progress), February 2016.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<http://www.rfc-editor.org/info/rfc6241>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<http://www.rfc-editor.org/info/rfc6991>>.
- [RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <<http://www.rfc-editor.org/info/rfc7223>>.
- [RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", RFC 7224, DOI 10.17487/RFC7224, May 2014, <<http://www.rfc-editor.org/info/rfc7224>>.
- [RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <<http://www.rfc-editor.org/info/rfc7277>>.
- [RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <<http://www.rfc-editor.org/info/rfc7317>>.
- [RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <<http://www.rfc-editor.org/info/rfc7752>>.



Authors' Addresses

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

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

Joel Halpern  
Ericsson

Email: [Joel.Halpern@ericsson.com](mailto:Joel.Halpern@ericsson.com)

Susan Hares  
Huawei

Email: [shares@ndzh.com](mailto:shares@ndzh.com)

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

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

Thomas D. Nadeau  
Brocade

Email: [tnadeau@lucidvision.com](mailto:tnadeau@lucidvision.com)

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: November 11, 2016

A. Atlas, Ed.  
Juniper Networks  
T. Nadeau, Ed.  
Brocade  
D. Ward  
Cisco Systems  
May 10, 2016

Interface to the Routing System Problem Statement  
draft-ietf-i2rs-problem-statement-11

Abstract

Traditionally, routing systems have implemented routing and signaling (e.g. MPLS) to control traffic forwarding in a network. Route computation has been controlled by relatively static policies that define link cost, route cost, or import and export routing policies. With the advent of highly dynamic data center networking, on-demand WAN services, dynamic policy-driven traffic steering and service chaining, the need for real-time security threat responsiveness via traffic control, and a paradigm of separating policy-based decision-making from the router itself, requirements have emerged to more dynamically manage and program routing systems. These requirements should allow controlling routing information and traffic paths and extracting network topology information, traffic statistics, and other network analytics from routing systems.

This document proposes meeting this need via an Interface to the Routing System (I2RS).

Status of This Memo

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

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

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

This Internet-Draft will expire on November 11, 2016.

## Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	2
2. I2RS Model and Problem Area for the IETF . . . . .	3
3. Standard Data-Models of Routing State for Installation . . . . .	6
4. Learning Router Information . . . . .	6
5. Aspects to be Considered for an I2RS Protocol . . . . .	7
6. Acknowledgements . . . . .	9
7. IANA Considerations . . . . .	9
8. Security Considerations . . . . .	9
9. References . . . . .	9
9.1. Normative References . . . . .	9
9.2. Informative References . . . . .	9
Appendix A. Existing Management Interfaces . . . . .	10
Authors' Addresses . . . . .	11

## 1. Introduction

Traditionally, routing systems have implemented routing and signaling (e.g. MPLS) to control traffic forwarding in a network. Route computation has been controlled by relatively static policies that define link cost, route cost, or import and export routing policies. With the advent of highly dynamic data center networking, on-demand WAN services, dynamic policy-driven traffic steering and service chaining, the need for real-time security threat responsiveness via traffic control, and a paradigm of separating policy-based decision-making from the router itself, the need has emerged to more dynamically manage and program routing systems in order to control routing information and traffic paths and to extract network topology information, traffic statistics, and other network analytics from routing systems.

As modern networks continue to grow in scale and complexity and desired policy has become more complex and dynamic, there is a need to support rapid control and analytics. The scale of modern networks and data-centers and the associated operational expense drives the need to automate even the simplest operations. The ability to quickly interact via more complex operations to support dynamic policy is even more critical.

In order to enable network applications to have access to and control over information in the different vendors' routing systems, a publicly documented interface is required. The interface needs to support real-time, asynchronous interactions using efficient data models and encodings that are based on and extend those previously defined. Furthermore, the interface must be tailored to provide a solid base on which a variety of use cases can be supported.

To support the requirements of orchestration software and automated network applications to dynamically modify the network, there is a need to learn topology, network analytics, and existing state from the network as well as to create or modify routing information and network paths. A feedback loop is needed so that changes made can be verifiable and so that these applications can learn and react to network changes.

Proprietary solutions to partially support the requirements outlined above have been developed to handle specific situations and needs. Standardizing an interface to the routing system will make it easier to integrate use of it into a network. Because there are proprietary partial solutions already, the standardization of a common interface should be feasible.

It should be noted that during the course of this document, the term "applications" is used. This is meant to refer to an executable program of some sort that has access to a network, such as an IP or MPLS network, via a routing system.

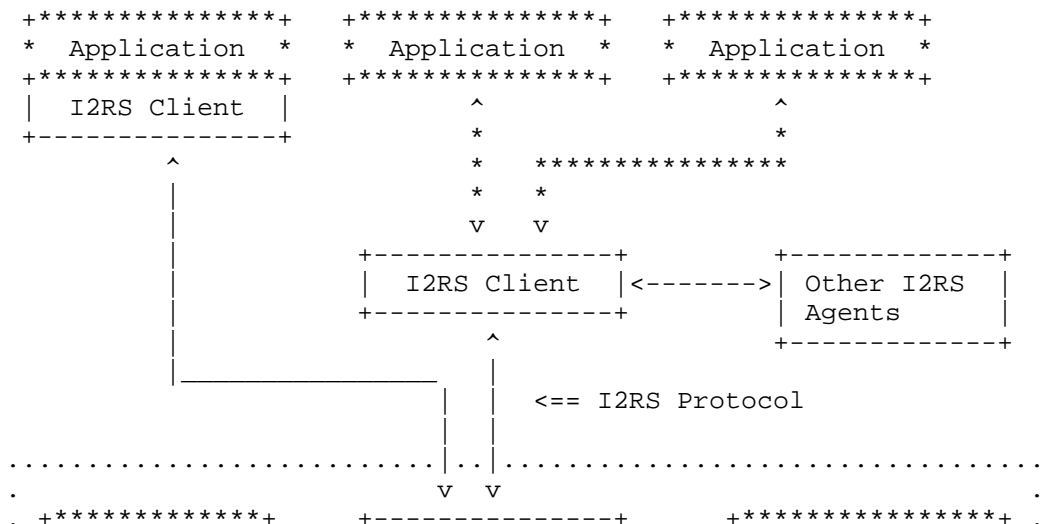
## 2. I2RS Model and Problem Area for the IETF

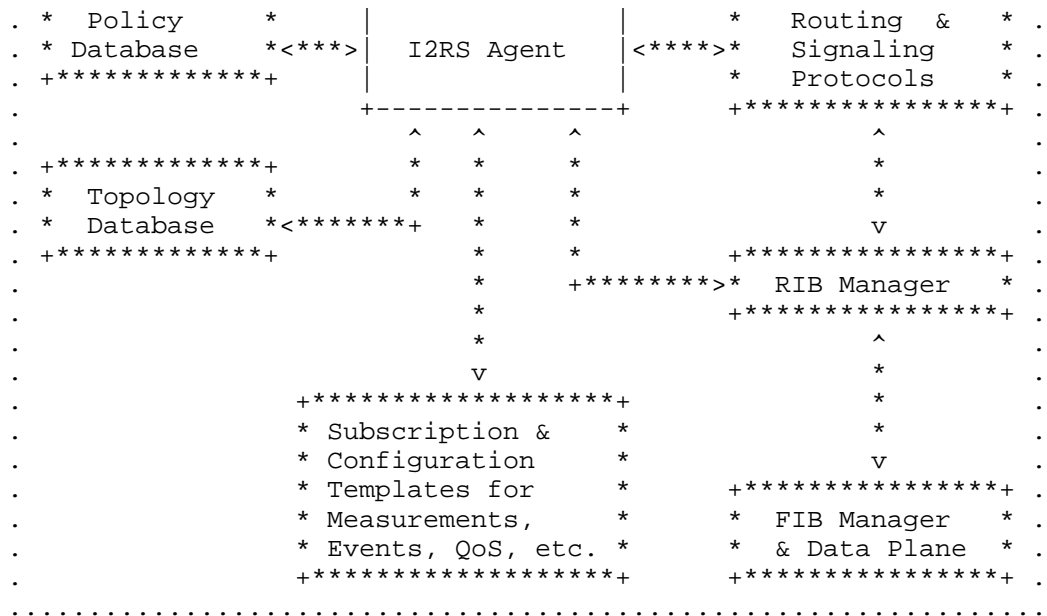
Managing a network of systems running a variety of routing protocols and/or providing one or more additional services (e.g., forwarding, classification and policing, firewalling) involves interactions between multiple components within these systems. Some of these systems or system components may be virtualized, co-located within the same physical system or distributed. In all cases, it is desirable to enable network applications to manage and control the services provided by many, if not all, of these components, subject to authenticated and authorized access and policies.

A data-model driven interface to the routing system is needed. This will allow expansion of what information can be read and controlled and allow for future flexibility. At least one accompanying protocol with clearly defined operations is needed; the suitable protocol(s) can be identified and expanded to support the requirements of an Interface to the Routing System (I2RS). These solutions must be designed to facilitate rapid, isolated, secure, and dynamic changes to a device's routing system. These would facilitate wide-scale deployment of interoperable applications and routing systems.

The I2RS model and problem area for IETF work is illustrated in Figure 1. This document uses terminology defined in [I-D.ietf-i2rs-architecture]. The I2RS Agent is associated with a routing element, which may or may not be co-located with a data-plane. The I2RS Client could be integrated in a network application or controlled and used by one or more separate network applications. For instance, an I2RS Client could be provided by a network controller or a network orchestration system that provides a non-I2RS interface to network applications and an I2RS interface to I2RS Agents on the systems being managed. The scope of the data-models used by I2RS extends across the entire routing system and the selected protocol(s) for I2RS.

As depicted in Figure 1, the I2RS Client and I2RS Agent in a routing system are objects with in the I2RS scope. The selected protocol(s) for I2RS extend between the I2RS client and I2RS Agent. All other objects and interfaces in Figure 1 are outside the I2RS scope for standardization.





<--> interfaces inside the scope of I2RS Protocol  
 +---+ objects inside the scope of I2RS-defined behavior  
  
 <\*> interfaces NOT within the scope of I2RS Protocol  
 +\*\*+ objects NOT within the scope of I2RS-defined behavior  
  
 <== used to point to the interface where the I2RS Protocol  
 would be used  
  
 .... boundary of a router supporting I2RS

Figure 1: I2RS model and Problem Area

The protocol(s) used to carry messages between I2RS Clients and I2RS Agents should provide the key features specified in Section 5.

I2RS will use a set of meaningful data-models for information in the routing system and in a topology database. Each data-model should describe the meaning and relationships of the modeled items. The data-models should be separable across different features of the managed components, versioned, and extendable. As shown in Figure 1, I2RS needs to interact with several logical components of the routing element: policy database, topology database, subscription and configuration for dynamic measurements/events, routing signaling

protocols, and its RIB manager. This interaction is both for writing (e.g. to policy databases or RIB manager) as well as for reading (e.g. dynamic measurement or topology database). An application should be able to combine data from individual routing elements to provide network-wide data-model(s).

The data models should translate into a concise transfer syntax, sent via the I2RS protocol, that is straightforward for applications to use (e.g., a Web Services design paradigm). The information transfer should use existing transport protocols to provide the reliability, security, and timeliness appropriate for the particular data.

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

As described in Section 1, there is a need to be able to precisely control routing and signaling state based upon policy or external measures. One set of data-models that I2RS should focus on is for interacting with the RIB layer (e.g. RIB, LIB, multicast RIB, policy-based routing) to provide flexibility and routing abstractions. As an example, the desired routing and signaling state might range from simple static routes to policy-based routing to static multicast replication and routing state. This means that, to usefully model next-hops, the data model employed needs to handle next-hop indirection and recursion (e.g. a prefix X is routed like prefix Y) as well as different types of tunneling and encapsulation.

Efforts to provide this level of control have focused on standardizing data models that describe the forwarding plane (e.g. ForCES [RFC3746]). I2RS recognizes that the routing system and a router's OS provide useful mechanisms that applications could usefully harness to accomplish application-level goals. Using routing indirection, recursion and common routing abstractions (e.g. tunnels, LSPs, etc.) provides significant flexibility and functionality over collapsing the state to individual routes in the FIB that need to be individually modified when a change occurs.

In addition to interfaces to control the RIB layer, there is a need to dynamically configure policies and parameter values for the various routing and signaling protocols based upon application-level policy decisions.

### 4. Learning Router Information

A router has information that applications may require so that they can understand the network, verify that programmed state is installed, measure the behavior of various flows, and understand the existing configuration and state of the router. I2RS should provide

a framework so that applications can register for asynchronous notifications and can make specific requests for information.

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

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

There are a wide range of events that applications could use for either verification of router state before other network state is changed (e.g. that a route has been installed), to act upon changes to relevant routes by others, or upon router events (e.g. link up/down). While a few of these (e.g. link up/down) may be available via MIB notifications today, the full range is not (e.g. route-installed, route-changed, primary LSP changed, etc.)

## 5. Aspects to be Considered for an I2RS Protocol

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

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

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

**Very Fine Granularity of Data Locking for Writing:** When an I2RS operation is processed, it is required that the data locked for writing is very granular (e.g. a particular prefix and route) rather than extremely coarse, as is done for writing



configuration. This should improve the number of concurrent I2RS operations that are feasible and reduce blocking delays.

**Multi-Headed Control:** Multiple applications may communicate to the same I2RS Agent in a minimally coordinated fashion. It is necessary that the I2RS Agent can handle multiple requests in a well-known policy-based fashion. Data written can be owned by different I2RS Clients at different times; data may even be overwritten by a different I2RS Client. The details of how this should be handled are described in [I-D.ietf-i2rs-architecture].

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

**High-Throughput:** At a minimum, the I2RS Agent and associated router should be able to handle a considerable number of operations per second (for example 10,000 per second to handle many individual subscriber routes changing simultaneously).

**Low-Latency:** Within a sub-second time-scale, it should be possible to complete simple operations (e.g. reading or writing a single prefix route).

**Multi-Channel:** It should be possible for information to be communicated via the interface from different components in the router without requiring going through a single channel. For example, for scaling, some exported data or events may be better sent directly from the forwarding plane, while other interactions may come from the control-plane. One channel, with authorization and authentication, may be considered primary; only an authorized client can then request that information be delivered on a different channel. Writes from a client are only expected on channels that provide authorization and authentication.

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

**Secure Control and Access:** Any ability to manipulate routing state must be subject to authentication and authorization. Sensitive routing information also may need to be provided via secure access

back to the I2RS Client. Such communications must be integrity protected. Most communications will also require confidentiality.

**Extensible and Interoperability:** Both the I2RS protocol and models must be extensible and interoperate between different versions of protocols and models.

## 6. Acknowledgements

The authors would like to thank Ken Gray, Ed Crabbe, Nic Leymann, Carlos Pignataro, Kwang-koog Lee, Linda Dunbar, Sue Hares, Russ Housley, Eric Grey, Qin Wu, Stephen Kent, Nabil Bitar, Deborah Brungard, and Sarah Banks for their suggestions and review.

## 7. IANA Considerations

This document includes no request to IANA.

## 8. Security Considerations

Security is a key aspect of any protocol that allows state installation and extracting of detailed router state. The need for secure control and access is mentioned in Section 5. More architectural security considerations are discussed in [I-D.ietf-i2rs-architecture]. Briefly, the I2RS Agent is assumed to have a separate authentication and authorization channel by which it can validate both the identity and the permissions associated with an I2RS Client. Mutual authentication between the I2RS Agent and I2RS Client is required. Different levels of integrity, confidentiality, and replay protection are relevant for different aspects of I2RS.

## 9. References

### 9.1. Normative References

[I-D.ietf-i2rs-architecture]  
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-15 (work in progress), April 2016.

### 9.2. Informative References

[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-13 (work in progress), October 2015.

- [RFC3746] Yang, L., Dantu, R., Anderson, T., and R. Gopal, "Forwarding and Control Element Separation (ForCES) Framework", RFC 3746, DOI 10.17487/RFC3746, April 2004, <<http://www.rfc-editor.org/info/rfc3746>>.
- [RFC4292] Haberman, B., "IP Forwarding Table MIB", RFC 4292, DOI 10.17487/RFC4292, April 2006, <<http://www.rfc-editor.org/info/rfc4292>>.
- [RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", RFC 5470, DOI 10.17487/RFC5470, March 2009, <<http://www.rfc-editor.org/info/rfc5470>>.

#### Appendix A. Existing Management Interfaces

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

There are three basic ways that routers are managed. The most popular is the command line interface (CLI), which allows both configuration and learning of device state. This is a proprietary interface resembling a UNIX shell that allows for very customized control and observation of a device, and, specifically of interest in this case, its routing system. Some form of this interface exists on almost every device (virtual or otherwise). Processing of information returned to the CLI (called "screen scraping") is a burdensome activity because the data is normally formatted for use by a human operator, and because the layout of the data can vary from device to device, and between different software versions. Despite its ubiquity, this interface has never been standardized and is unlikely to ever be standardized. CLI standardization is not considered as a candidate solution for the problems motivating I2RS.

The second most popular interface for interrogation of a device's state, statistics, and configuration is the Simple Network Management Protocol (SNMP) and a set of relevant standards-based and proprietary Management Information Base (MIB) modules. SNMP has a strong history of being used by network managers to gather statistical and state information about devices, including their routing systems. However, SNMP is very rarely used to configure a device or any of its systems for reasons that vary depending upon the network operator. Some example reasons include complexity, the lack of desired configuration semantics (e.g., configuration "roll-back", "sandboxing" or configuration versioning), and the difficulty of using the semantics

(or lack thereof) as defined in the MIB modules to configure device features. Therefore, SNMP is not considered as a candidate solution for the problems motivating I2RS.

Finally, the IETF's Network Configuration (or NETCONF) protocol has made many strides at overcoming most of the limitations around configuration that were just described. However, as a new technology and with the initial lack of standard data models, the adoption of NETCONF has been slow. I2RS will identify and define as needed information and data models to support I2RS applications. Additional extensions to handle multi-headed control may need to be added to NETCONF and/or appropriate data models.

#### Authors' Addresses

Alia Atlas (editor)  
Juniper Networks

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

Thomas D. Nadeau (editor)  
Brocade

Email: [tnadeau@lucidvision.com](mailto:tnadeau@lucidvision.com)

Dave Ward  
Cisco Systems

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

INTERNET-DRAFT  
Intended Status: Standard Track

Rex Fernando  
Sami Boutros  
Dhananjaya Rao  
Cisco Systems

Nabil Bitar  
Luay Jalil  
Verizon

Expires: January 7, 2016

July 6, 2015

Interface to a Packet Switching Element (IPSE)  
draft-rfernando-ipse-02.txt

#### Abstract

This document describes a set of mechanisms for decoupling the control and data plane of a routing or a switching device and describes an open API between the two that satisfies a set of constraints that are desirable for such an interface.

#### 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) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1	Introduction . . . . .	4
1.1	Terminology . . . . .	4
2.	Routing and Switching Systems Model . . . . .	4
2.1	Control Plane (Route Controller) . . . . .	5
2.2	Forwarding Plane (Packet Switching Element) . . . . .	5
2.3	Motivation for separation of Route Controller and Packet Switching Elements(s) . . . . .	5
2.4	Different constructions of RC and PSE(s) . . . . .	7
3.	Packet Switching Element Conceptual Model . . . . .	8
3.1	Typical PSE model . . . . .	8
3.1.1	I/P Block (Input Processing) . . . . .	8
3.1.2	O/P Block (Output Processing) . . . . .	9
3.1.3	Forwarding Block . . . . .	9
3.1.3.1	Forwarding block model . . . . .	9
3.1.3.2	Context Selection Block . . . . .	10
4.	Yang modules . . . . .	11
4.1	YANG Module "pse" . . . . .	11
4.2	YANG Module "pse-common-types" . . . . .	16
4.3	Global sub-modules of "pse" . . . . .	17
4.3.1	YANG sub module "pse-oam" . . . . .	17
4.3.2	YANG submodule "physical-interfaces" . . . . .	19
4.3.3	YANG submodule "context-selector-table" . . . . .	20
4.3.4	YANG submodule "label-table" . . . . .	22
4.3.5	YANG submodule "l2tp-table" . . . . .	26
4.4	PER Tenant submodules . . . . .	27
4.4.1	YANG submodule "ip-unicast-table" . . . . .	27
4.4.2	YANG submodule "flow-table" . . . . .	30
4.4.3	YANG module "ip-next-hop" . . . . .	30
4.4.4	YANG submodule "l2-table" . . . . .	32
4.4.5	YANG submodule "l2-next-hop" . . . . .	33
4.4.6	YANG submodule "interface-table" . . . . .	36

4.4.7 YANG submodule "arp-table" . . . . .	39
4.4.8 Yang submodule "arp-proxy-table" . . . . .	40
5 Performance and scale requirements . . . . .	41
6 YANG data model tree . . . . .	42
7 Major Contributing Authors . . . . .	47
8 Security Considerations . . . . .	47
9 IANA Considerations . . . . .	47
10 References . . . . .	47
10.1 Normative References . . . . .	47
10.2 Informative References . . . . .	47
Authors' Addresses . . . . .	47

## 1 Introduction

This document describes a data model driven API to program a routing and switching system's forwarding plane. It describes the motivations for creating an open API between a router's control and its forwarding plane and lays out the exact mechanisms that can be used by the control plane of a routing system to "program" the forwarding tables and state contained in the forwarding plane and additionally get notifications when that state changes in interesting ways. It also describes APIs to extract information from the forwarding plane, such as queue statistics.

The document proposes YANG as the modeling language for this purpose and provides the exact models that represent the forwarding state of a routers data plane.

YANG provides Hierarchical data models and modularity through the use of modules and submodules, that we can use to express the forwarding tables, forwarding objects and adjacencies. The ease of extending the data model through augmentation mechanisms and the versioning rules supported by YANG would allow us to easily extend the model with new forwarding features.

The use of YANG would allow us to decouple the data model from the encoding method and the transport protocol. The encoder/decoder for YANG is agnostic of the forwarding data model specifics.

Netconf has been used as the transport protocol for YANG, and Netconf is a socket based reliable transport protocol that defines as well security and authentication semantics for the communication of the YANG data model if required.

Note that this document uses the word "router" to refer to any network packet forwarding device. The mechanisms outlined here are equally applicable to L3 devices (routers), L2 devices (switches), MPLS label switches and any device that's a hybrid.

### 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. Routing and Switching Systems Model



Most modern routing and switching systems have two main components: the control plane and the data plane. Traditionally these systems implement proprietary closed communication between the control and the data plane.

## 2.1 Control Plane (Route Controller)

The control plane typically implements different functions on a routing and switching system, which include open interfaces for external management and orchestration, an interface to retrieve operational data, routing and switching control protocols to interact with other network elements or hosts, route computation and database to store computed results (RIB) and internal event handling infrastructure that glues these control components together.

The rest of the document will refer to the control plane as the "route controller" or just RC.

## 2.2 Forwarding Plane (Packet Switching Element)

The forwarding plane performs the forwarding functions on packets received on physical interfaces as per the control directives and state added by the control plane. It also informs the control plane of local events and various operational state so that the control plane can take the necessary actions.

The rest of this document will refer to the forwarding plane as the "packet switching element" or just PSE.

The PSE has the following functional modules:

1. An PSE control agent that interfaces with the control plane. The agent supports the protocol that is used to communicate with the route controller. It processes the forwarding table and interface updates sent by the controller and populates the forwarding tables policy rules and associated actions that the packet processing logic needs. It also reports events and statistics to the route controller
2. Packet processing module that performs the actual packet forwarding and packet manipulation actions. It has a set of forwarding and interface tables that it uses to lookup both runtime and configuration state. The packet processing module also collects various statistics and error counters, as well as generates events.

## 2.3 Motivation for separation of Route Controller and Packet Switching Elements(s)

The advent of software defined networking (SDN) and virtualization have opened up many well understood reasons for creating an open interface between the route controller (RC) and forwarding element (PSE). The reasons include but not limited to such factors as, (a) being able to run the route controller on commodity x86 servers and still be able to control the data plane of legacy devices, (b) being able to create interesting routing and bridging topologies and realizing policies at the forwarding plane, (c) modularizing the interface allows best in class products to be assembled without a vendor "lock-in" and the innovation that such a model provides and (d) the cost savings provided by the de-coupled model where a central control plane can provide better control and efficiency in directing packet flows than the traditional distributed protocol based control plane and (e) the ability to scale out the control plane independently from the forwarding plane, maximizing resource utilization. Specifically, a control plane can scale out to control a large number of forwarding modules that are part of the same routing system while maximizing the utilization of the forwarding modules' hardware resources

In short the benefits of software driven networking can be effectively realized by decoupling the route controller and packet switching elements and providing an open interface between the two.

An important consideration in providing this modularity is the 'reuse' of technology concepts instead of reinventing a brand new mechanism at the data plane level just to provide an open interface. There are many downsides to creating a new data plane model than the well understood 'IP routing' or 'L2 switching' models - the foremost of them being that existing equipment where the hardware behavior is 'baked' in have to be forklift upgraded and replaced with devices that support the new forwarding model. Even if one were to ignore the impracticality of such a solution, one cannot ignore the investment loss in rendering all the legacy devices useless. Let alone that it might toe the innovation in implementations. There is need to abstract the actual forwarding model implementation from the control applications that can manipulate the behavior of the forwarding plane or extract information from it.

A more practical approach to solving the control and data plane separation is to provide an interface to these devices that use existing data plane concepts and objects such as IP prefixes, L2 mac entries, next-hops (direct and recursive), vlans, MPLS labels, tunnel encapsulations, interfaces/ports, ECMP forwarding and VRF, EVI and VSI containers, and policies and associated actions. Hence, IPSE expresses the forwarding data model as a chain of forwarding objects, and allowing updates to any object at any level of the chain.

Most devices understand these concepts and objects and use these concepts and objects to classify and forward data traffic. The idea behind IPSE is to create an abstraction layer to these forwarding objects that allows an external control plane to communicate with a forwarding plane to program these objects. The packet switching element (PSE) itself could be both hardware or software based.

Software based PSE's are increasingly becoming popular especially in data center and cloud use cases where multi-tenanted topologies can be rapidly created by installing a software switch/router in the server hosting the cloud and data center applications and using a central controller to program their forwarding tables [VPE DRAFT].

#### 2.4 Different constructions of RC and PSE(s)

While the RC and PSEs can be separated from a communication perspective as outlined in the previous section, there are two distinct ways they can be packaged to work with each other.

The more traditional packaging involves both these functions co-located in a single device (for instance within a router chassis). The de-coupling of the controller and PSE enables them to be separated out with the RC running in a different device than the PSE. This might be desirable, because, the controller as an application has different functional, performance and scale characteristics that it would be inefficient and suboptimal for it to be coupled with the forwarding elements.

One of the benefits of the model based approach to the separation of RC and PSE is that, as long as the PSE(s) implements the models described in this document and can have a back-end that can honor the semantics of the models defined here, the PSE could be either software based or hardware based. In addition, how the model translates into implementation is abstracted away.

This allows for integration of not just software based forwarding elements but can also include hardware forwarding elements (legacy router forwarding plane) as long as the forwarding plane abstraction layer implements the YANG models defined in this document. <The specific hardware mechanisms used to implement the models may vary depending on the device.>

Lastly, when it comes to routing and reachability, the approach we take to scale the system is to treat the RC and PSE(s) as have been viewed traditionally. IOW, a set of PSE(s) can be programmed by a RC. An RC could be cluster of two or servers, each server containing one or more CPU, associated memory, storage and network interfaces. A

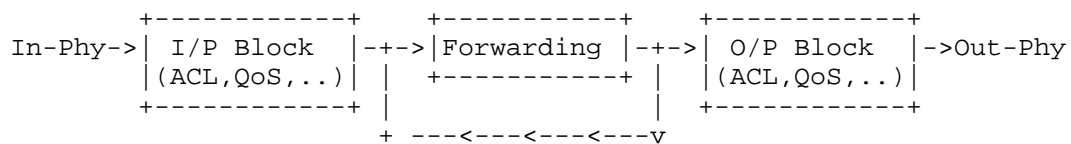
routing system will be composed of an RC (and optionally a redundant RC) and a set of PSEs. Two such routing systems would then talk to each other using traditional routing and signaling protocols as needed, and to hosts or host proxies using traditional protocols (e.g., DHCP). On the opposite, the proposed architecture is targeted to better enable scalability by breaking the traditional topology where the RC is boxed in with the a set of PSEs in the same chassis. This alignment to an existing routing model also allows one to keep certain operational aspects intact with respect to the RC/PSE construction. When properly built, the fact that RC and PSE(s) are de-coupled may not be exposed to operational systems if needed.

### 3. Packet Switching Element Conceptual Model

This section describes in brief the conceptual model for the packet switching element (PSE). It describes the various functional blocks and typical forwarding chains that a packet may pass through within the PSE. The various tables specified in the data model in the subsequent section contain the state needed to carry out the packet processing described here.

#### 3.1 Typical PSE model

In the packet forwarding path, an PSE consists of the following high-level logical blocks.



In-Phy: A Physical interface via which packets are received by the device.

Out-Phy: A physical interface via which packets are transmitted out of the device.

##### 3.1.1 I/P Block (Input Processing)

This is a logical packet processing block at the input to a routing or switching device. This block acts upon the packet based on the interface properties and ingress policing rules and packet attributes. It also generates the necessary inputs to the forwarding block.

A device contains two broad categories of interfaces - those facing

the edge or providing access to an edge network or device, and those facing the core network. The input (and output) processing logic applies to both categories, though the rules and actions performed might be different.

### 3.1.2 O/P Block (Output Processing)

This is a logical packet processing block at the output stage of packet forwarding. This block typically applies the output features such as output port policing, queuing and filtering before transmitting the packet out on the physical port.

### 3.1.3 Forwarding Block

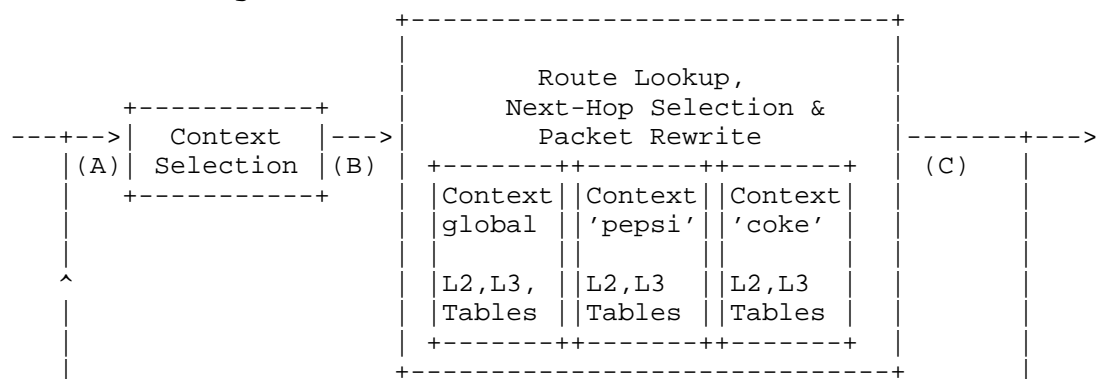
This is the block that contains the various forwarding and adjacency tables, and the logic that does the forwarding table lookups and packet rewrites. This document focuses on the models required to describe the behavior and state required by the forwarding block.

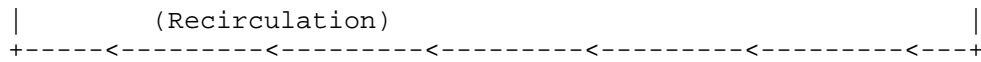
Depending on the type of an incoming packet, it is possible to do forwarding based on different attributes or combinations of attributes in the packet header. Each of these attributes can be considered as a forwarding class, and a given packet may be subject to multiple of them, based on user requirements.

The forwarding-class selector logic identifies the fields in the packet header that will be used to do the lookup for each forwarding-class supported on the device. The lookup occurs in a specific table for that class.

There is a logically separate FIB table for each class. Additionally, a device may support multiple logical instances of each table, for example, an IP FIB per VRF.

#### 3.1.3.1 Forwarding block model





#### Forwarding Block

Conceptually, there are two components to the forwarding block. The context selection logic and the actual route lookup, next-hop selection and packet rewrite block. The idea is that an incoming packet can be processed completely by these two blocks and then sent out on an output port towards its next-hop. In some cases, however, it might be easier to construct more complex scenarios by recirculating the packet multiple times through these two blocks.

For instance, when an MPLS-VPN packet is received in the core facing interface, the packet is encapsulated with the VPN label. The VPN label is looked up in the global label table, the result of the lookup could point to the final next-hop or to the context ip table in which the ip packet destination should be looked up in to find the final next-hop. It is easier to construct this 'chained' operation by simply recirculating the packet through the 'context selector' and 'route lookup' blocks than constructing a specific chain for this particular scenario.

The idea then is to generalize the forwarding actions by decomposing them into a series of 'context selection' followed by 'route lookup' operations than creating static chain of such operations on a per scenario basis.

#### 3.1.3.2 Context Selection Block

The context-selector block determines the specific forwarding context and the specific table within that context in which to perform the forwarding lookup for an incoming packet. A device may have multiple forwarding contexts to support multi-tenancy. Each forwarding context could have multiple tables to support different types of lookups (L2, IPv4, IPv6, flow, etc).

Inputs to the context selection is the received packet itself together with information on the input interface (physical or logical) on which the packet was received.

Currently the model supports determination of the context and table based on the switching mode set on the interface or based on the table pointed to by the MPLS VPN label on the packet.

The output of the context selection block is the {context, table}

pair to perform the lookup on. The output table could be L2, IPv4 or IPv6 based on the packet type and the mode set on the input interface or as implied by the VPN label on the packet.

#### 4. Yang modules

In this section we describe the different YANG modules implementing the data model for the Packet Switching Elements (PSEs). The following sections will define a hierarchy of tables that are needed to implement the L2, L3, MPLS and flow forwarding functions.

As is evident, there are cases where the forwarding lookup has to be scoped by a given 'tenant'. In other cases the lookup has to be performed at a global level.

At the top level we will define tables that have global scope i.e. not per tenant. The following are the tables that have global scope,

- a. The interface table that describe the interfaces attached to the PSE's.
- b. The Context Selector table that tells the PSE how to map traffic arriving on a physical or logical interface to the tenant L2, L3 or flow forwarding tables. The output of the context selector table is a tenant context and a table name within that context to perform a lookup on.
- c. The MPLS label table that lists all the MPLS labels and the associated tenant L2, L3 of flow forwarding tables
- d. The L2 tunnel table that will list all the tenants L2 tunnels for point2point cross-connects
- e. The OAM table that lists the logical or physical interfaces as well as the end-point prefixes that need to be monitored by the forwarding element. The interface state is monitored through local events such as 'link up' and 'link down'. Prefixes are monitored through OAM protocols such as ICMP or BFD. A change in state of interfaces or prefix reachability could trigger notification to the route controller to ensure service assurance and continuity.

Then we will be defining the per tenant forwarding table hierarchy, in which we will define both L2, L3 and flow forwarding tables and the corresponding L2 and L3 next-hops.

##### 4.1 YANG Module "pse"

The Packet Switching Element (PSE) module is the top level module

that allows for provisioning and managing a set of global tables and tenant specific tables. The module defines read-only variables that allows the route controller to query for PSE's forwarding capabilities and features that are supported by the PSE that can be turned on or utilized by the RC.

It allows for provisioning for IPv4, IPv6, L2, MPLS Label & Flow forwarding tables based on device capabilities.

As the top level table, it includes other tables (such as interface table, arp table, L3 and L2 tables) that are needed to perform packet forwarding.

```
module pse {
  namespace "http://www.cisco.com/yang-modules/ipse/pse";
  prefix "pset";
  import ietf-inet-types {
    prefix "inet";
  }

  import pse-common-types {
    prefix "ct";
  }

  include interface-table;
  include context-selector-table;
  include arp-table;
  include arp-proxy-table;
  include l2tp-table;
  include pse-oam;
  include ip-unicast-table;
  include l2-table;
  include flow-table;
  include label-table;

  description "PSE stands for Packet Switching Element.
    The module allows for provisioning and managing
    multi-tenant forwarding tables in a network device.
    Currently supports IPv4, IPv6, L2 & Flow forwarding
    entries based on device capabilities. This module will be
    extended to support multicast in future";

  organization "Cisco Systems";
  contact "employee@cisco.com";

  revision 2013-12-05 {
    description "Initial revision.";
```



```
}  
  
container pse-global-features {  
    description "PSE capabilities and features (read only)";  
    config false;  
  
    leaf ready {  
        description "Once the controller connects to the pse  
            it can be notified (or it can query) about the pse  
            state to check if pse is READY";  
  
        type boolean;  
    }  
  
    leaf ip-unicast-forwarding {  
        description "This feature indicates that the device  
            implements IPv4 based unicast forwarding";  
  
        type boolean;  
    }  
  
    leaf l2-forwarding {  
  
        description "This feature indicates that the device  
            implements forwarding based on L2 addresses";  
  
        type boolean;  
    }  
  
    leaf flow-forwarding {  
  
        description "This feature indicates that the device  
            implements flow forwarding based on (src-ip, dst-ip,  
            src-port, dst-port, protocol)";  
  
        type boolean;  
    }  
  
    uses ip-unicast-features;  
    uses l2-features;  
    uses flow-features;  
}  
  
container pse-oam {  
  
    description "This global table can be used to determine  
        the health status of objects - such as interface state  
        and reachability of a prefix";
```

```
    uses pse-oam;
}

container sync {
    leaf sync-state {
        description "Used to Synchronize PSE tables. These
            variables could be used by the route controller to
            indicate the start and end of a route download
            operation";

        type enumeration {
            enum sync-start;
            enum sync-complete;
        }
    }
}

container context-selector-table {
    uses context-selector-table;
}

uses label-table;

container interface-table {
    uses if-table;
}

container protocol-addresses {

    container ipv4 {
        container table {
            uses ct:table-name;
        }
        leaf source-address {
            type inet:ipv4-address;
        }
        leaf dhcp-address {
            type inet:ipv4-address;
        }
        leaf gateway-address {
            type inet:ipv4-address;
        }
    }

    container ipv6 {
        container table {
            uses ct:table-name;
        }
    }
}
```

```
        leaf source-address {
            type inet:ipv6-address;
        }
        leaf dhcp-address {
            type inet:ipv6-address;
        }
        leaf gateway-address {
            type inet:ipv6-address;
        }
    }
}

uses l2tp-table;

list pse-contexts {

    description "This list represents a provisioned list of
    packet forwarding contexts. Each context could represent
    a distinct tenant and network address space. Forwarding
    tables and other policy parameters related to routing,
    addressing, forwarding are assumed to be defined within
    a pse context. Each pse context is uniquely identified by
    a name";

    key pse-context-name;

    leaf pse-context-name {
        type string;
    }

    list tables {
        description
            "list of forwarding tables within the context,
            indexed by unique table's name ";

        key table-name;

        leaf table-name {
            type string;
            description "Instance of a forwarding table
            identified by name";
        }

        leaf pse-vpn-id {
            type string;
            description "VPN-ID used by DHCP";
        }
    }
}
```

```

        choice pse-table-type {
            case ip-unicast-table {

                container arp-table {
                    uses arp-table;
                }

                container arp-proxy-table {
                    uses arp-proxy-table;
                }

                container ip-unicast-table {
                    uses ip-unicast-table;
                }
            }

            case l2-table {
                container l2-table {
                    uses l2-table;
                }
            }

            case flow-table {
                container flow-table {
                    uses flow-table;
                }
            }
        }
    }
}

```

#### 4.2 YANG Module "pse-common-types"

This module defines some common types used by all the modules and sub modules defined in this document.

```

module pse-common-types {

    namespace
    "http://www.cisco.com/yang-modules/ipse/pse-common-types";
    prefix "psect";

    revision "2013-12-05" {
        description "Initial revision.";
    }

    grouping table-name {

```

```
description "The unique table's name is specified by PSE
context-name, and the table's name within the context";

leaf pse-ctx-name {
    description "The name of PSE context";
    type string;
}

leaf ctx-tbl-name {
    description "The name of table within PSE context";
    type string;
}
}

typedef interfaceName {
    type string;
}

typedef interfaceLogicalUnit {
    type int32 {
        range "0..9999";
    }
}

typedef prefixLengthIPv4 {
    type int32 {
        range "0..32";
    }
}

typedef prefixLengthIPv6 {
    type int32 {
        range "0..128";
    }
}
}
```

#### 4.3 Global sub-modules of "pse"

##### 4.3.1 YANG sub module "pse-oam"

This sub module defines the provisioned and operational list of interfaces and prefixes to be monitored by the devices implementing the PSE.

```
submodule pse-oam {
    belongs-to pse {
```

```
    prefix "pset";
}

import pse-common-types {
    prefix "ct";
}

import ietf-inet-types {
    prefix "inet";
}

organization "Cisco Systems";
contact "joe@acme.example.com";
description
    "OAM Status for PSE ";
revision 2013-12-05 {
    description "Initial revision.";
}

grouping pse-oam {
    container prefix-table {
        list unicast-prefix {
            description
                "This list defines the prefixes to be monitored by
                OAM.";
            key "prefix";

            leaf prefix {
                type inet:ip-prefix;
            }
        }
    }

    container interface-table {
        list interfaces {
            description
                "This list defines the interfaces to be monitored by
                OAM";

            key "interface-name";

            leaf interface-name {
                type ct:interfaceName;
            }
        }
    }
}
```

```

    container oper-prefix-down-table {
        config false;

        list oper-prefix-down-list {
            description
                "This list defines the prefixes oper down state
                monitored by OAM";

            key "version";

            leaf version {
                description "Version ";
                type uint32;
            }

            leaf prefix {
                type inet:ip-prefix;
            }
        }
    }

    container oper-interface-down-table {

        config false;

        list oper-interfaces-down-list {
            description
                "This list defines the interfaces oper down
                state monitored by OAM";

            key "version";

            leaf version {
                description "Version ";
                type uint32;
            }

            leaf interface-name {
                type ct:interfaceName;
            }
        }
    }
}

```

#### 4.3.2 YANG submodule "physical-interfaces"

This submodule defines the list of physical interfaces discovered on

the devices implementing the PSE.

```
submodule physical-interfaces {  
    belongs-to pse-tables {  
        prefix "pset";  
    }  
  
    import pse-common-types {  
        prefix "ct";  
    }  
  
    organization "Cisco Systems";  
    contact "employee@cisco.com";  
  
    description  
        "The module list the operational physical interfaces";  
  
    revision 2013-12-05 {  
        description "Initial revision.";  
    }  
  
    grouping physical-interfaces {  
        container physical-interfaces {  
            config false;  
            list interfaces {  
                description  
                    "This list define the physical interfaces discovered  
                    on end server";  
                key "interface-name";  
  
                leaf interface-name {  
                    type ct:interfaceName;  
                }  
            }  
        }  
    }  
}
```

#### 4.3.3 YANG submodule "context-selector-table"

This submodule defines the list of per tenant selector types.

```
submodule context-selector-table {  
    belongs-to pse-tables {  
        prefix "pset";  
    }  
}
```



```
import pse-common-types {
    prefix "ct";
}

organization "Cisco Systems";
contact "joe@acme.example.com";

description
    "The module maps incoming traffic to the forwarding table";

revision 2013-12-05 {
    description "Initial revision.";
}

typedef tbl-selection-type {
    type enumeration {
        enum ipv4 {
            description "The table for IPv4 traffic is derived
                        from interface configuration";
        }
        enum ipv6 {
            description "The table for IPv6 traffic is derived
                        from interface configuration";
        }
        enum l2-table {
            description "The table for L2 lookup is derived
                        from interface configuration";
        }
        enum flow-table {
            description "The table for flow based forwarding is
                        derived from interface configuration";
        }
    }
}

grouping context-selector-table {
    container label-ctx-selection-table {
        list label-entry {
            key label;
            leaf label {
                type uint32;
            }

            container disposition-table {
                uses ct:table-name;
            }
        }
    }
}
```

```

    container interface-ctx-selector-table {
      list interfaces {
        description
          " This list define the table selection mechanism
            on interface";

        key "interface-name selection-type";
        leaf interface-name {
          type ct:interfaceName;
        }

        leaf selection-type {
          type tbl-selection-type;
        }

        choice tbl-selection-type {
          case ipv4 {
            container ipv4-table {
              uses ct:table-name;
            }
          }
          case ipv6 {
            container ipv6-table {
              uses ct:table-name;
            }
          }
          case l2-table {
            container ipv6-table {
              uses ct:table-name;
            }
          }
          case flow-table {
            container ipv6-table {
              uses ct:table-name;
            }
          }
        }
      }
    }
  }
}

```

#### 4.3.4 YANG submodule "label-table"

This submodule defines the list of MPLS labels pointing to the different per tenant L2 and L3 tables.

```

submodule label-table {

```

```
belongs-to pse-tables {
    prefix "pset";
}

import ietf-inet-types {
    prefix "inet";
}

import ietf-yang-types {
    prefix "yang";
}

import pse-common-types {
    prefix "ct";
}

organization "Cisco Systems";
contact "joe@acme.example.com";

description
    "The module defines a forwarding tables based on IPv4/V6
    addresses";

revision 2013-12-05 {
    description "Initial revision.";
}

grouping out-label-stats {
    leaf packets {
        type yang:counter64;
    }

    leaf bytes {
        type yang:counter64;
    }
}

grouping mpls-next-hop {
    choice ip-next-hop-type {
        case ip-nh-type {
            description
                "Specifies the content of unicast IP next-hop";
            leaf ip-next-hop {
                type inet:ip-address;
            }

            leaf interface {
```

```
        type ct:interfaceName;
    }

    container nh-tbl-name {
        uses ct:table-name;
    }
}

case ip-nh-type-mpls-gre {
    leaf local-gre-addr {
        type inet:ip-address;
    }
    leaf remote-gre-addr {
        type inet:ip-address;
    }
    container nh-gre-tbl-name {
        uses ct:table-name;
    }

    leaf mpls-label {
        type uint32;
    }

    leaf gre-key {
        type uint32;
    }
}

case ip-nh-type-vxlan {
    leaf local-vxlan-addr {
        type inet:ip-address;
    }

    leaf remote-vxlan-addr {
        type inet:ip-address;
    }

    leaf vxlan-id {
        type uint32;
    }
}

leaf label {
    description "output label";
    type uint32;
}
```

```
    leaf label-type {
      type enumeration {
        enum valid {
          description "valid output label";
        }
        enum pop {
          description "pop-and-forward";
        }
      }
    }

    container next-hop-stats {
      config false;
      uses out-label-stats;
    }
  }

  grouping mpls-nh-array {
    list mpls-nh-array {
      key next-hop-index;
      leaf next-hop-index {
        type uint8;
        description "The numerical index of next-hop in
                     next-hop array";
      }

      leaf nh-weight {
        type uint32;
      }

      uses mpls-next-hop;
    }
  }

  typedef LocalLabelType {
    type enumeration {
      enum head-prefix {
        description "LSP head per-prefix label";
      }
      enum label-xconnect {
        description "Midpoint label";
      }
    }
  }

  grouping local-label-attributes {
    leaf local-label-type {
      type LocalLabelType;
    }
  }
```

```

        description "label type";
    }

    choice local-label-type-choice {
        case head-prefix {
            leaf prefix {
                type inet:ip-prefix;
            }
            container pref-table {
                uses ct:table-name;
            }
        }
    }
}

grouping label-table {
    list label-table {
        key "local-label";

        leaf local-label {
            description "input label";
            type uint32;
        }

        container label-attributes {
            uses local-label-attributes;
        }

        uses mpls-nh-array;
    }
}

```

#### 4.3.5 YANG submodule "l2tp-table"

This submodule defines the list of per tenant layer 2 tunnel protocol (l2tp) point2point cross connects.

```

submodule l2tp-table {

    belongs-to pse-tables {
        prefix "pset";
    }

    import ietf-inet-types {
        prefix "inet";
    }
}

```

```
import pse-common-types {
    prefix "ct";
}

organization "Cisco Systems";
contact "joe@acme.example.com";

description
    "L2TP v3/v6 tables";

revision 2013-12-05 {
    description "Initial revision.";
}

grouping l2tp-table {
    list l2tp {
        key "src-addr dst-addr session-id";

        leaf src-addr {
            type inet:ip-address;
        }

        leaf dst-addr {
            type inet:ip-address;
        }

        leaf session-id {
            type uint32;
        }

        leaf src-cookie {
            type uint32;
        }

        leaf dst-cookie {
            type uint32;
        }

        leaf interface-name {
            type ct:interfaceName;
        }
    }
}
```

#### 4.4 PER Tenant submodules

##### 4.4.1 YANG submodule "ip-unicast-table"

This submodule defines the per tenant ipv4/v6 forwarding tables.

```
submodule ip-unicast-table {  
    belongs-to pse-tables {  
        prefix "pset";  
    }  
  
    import ietf-yang-types {  
        prefix "yang";  
    }  
  
    import iana-afn-safi {  
        prefix "ianaaf";  
    }  
  
    import ietf-inet-types {  
        prefix "inet";  
    }  
  
    import ip-next-hop {  
        prefix "ipnh";  
    }  
    organization "Cisco Systems";  
    contact "joe@acme.example.com";  
  
    description  
        "The module defines a forwarding tables based on IPv4/V6  
        addresses";  
  
    revision 2013-12-05 {  
        description "Initial revision.";  
    }  
  
    grouping ip-unicast-features {  
        container ipv4-unicast-features {  
            description "This feature indicates that the device  
                implements IPv4 based unicast forwarding";  
  
            container ipv4-features {  
                uses ipnh:ip-next-hop-features;  
            }  
        }  
  
        container ipv6-unicast-features {  
            description "This feature indicates that the device
```



```
        implements IPv6 based unicast forwarding";

        container ipv6-features {
            uses ipnh:ip-next-hop-features;
        }
    }

    grouping ip-tbl-attribute {
        leaf afi {
            description "Type of IP table: ipv4/ipv6";
            type ianaaf:address-family;
        }
    }

    grouping prefix-stats {
        leaf packets {
            type yang:counter64;
        }

        leaf bytes {
            type yang:counter64;
        }
    }

    grouping ip-unicast-table {
        container ip-tbl-attrs {
            uses ip-tbl-attribute;
        }

        list unicast-prefix {
            key "prefix";

            leaf prefix {
                type inet:ip-prefix;
            }

            uses ipnh:ip-nh-array;

            container per-prefix-stats {
                config false;
                uses prefix-stats;
            }
        }
    }
```

```
    }  
}
```

#### 4.4.2 YANG submodule "flow-table"

This submodule defines the per tenant flow tables to allow for per flow forwarding across devices implementing the PSE.

TBD.

#### 4.4.3 YANG module "ip-next-hop"

This submodule defines per tenant the list of different L3 next hops that can be pointed at by L2 and L3 prefixes in the L2 or L3 tables.

```
module ip-next-hop {  
  
    namespace  
    "http://www.cisco.com/something/something-else/context1";  
    prefix "ipnh";  
  
    import ietf-inet-types {  
        prefix "inet";  
    }  
  
    import pse-common-types {  
        prefix "ct";  
    }  
  
    description  
        "The module defines a set of next-hop tables  
        specific forwarding tables can be derived";  
  
    revision 2013-12-05 {  
        description "Initial revision.";  
    }  
  
    grouping ip-next-hop-features {  
        leaf ecmp-supported {  
            type boolean;  
        }  
  
        leaf ucmp-supported {  
            type boolean;  
        }  
  
        leaf max-mp-next-hops {
```

```
        type uint32;
    }

    container nh-type-supported {
        leaf nh-type-regular {
            type boolean;
        }

        leaf nh-type-mpls-over-gre {
            type boolean;
        }

        leaf nh-type-vxlan {
            type boolean;
        }
    }
}

grouping ip-next-hop {
    choice ip-next-hop-type {

        case ip-nh-type {
            description
                "Specifies the content of unicast IP next-hop";
            leaf ip-next-hop {
                type inet:ip-address;
            }

            leaf interface {
                type ct:interfaceName;
            }

            container nh-tbl-name {
                uses ct:table-name;
            }
        }

        case ip-nh-type-mpls-gre {
            leaf local-gre-addr {
                type inet:ip-address;
            }

            leaf remote-gre-addr {
                type inet:ip-address;
            }

            container nh-gre-tbl-name {
                uses ct:table-name;
            }
        }
    }
}
```

```

    }

    leaf mpls-label {
        type uint32;
    }

    leaf gre-key {
        type uint32;
    }
}

case ip-nh-type-vxlan {
    leaf local-vxlan-addr {
        type inet:ip-address;
    }

    leaf remote-vxlan-addr {
        type inet:ip-address;
    }

    leaf vxlan-id {
        type uint32;
    }
}
}

grouping ip-nh-array {
    list ip-nh-array {
        key next-hop-index;
        leaf next-hop-index {
            type uint8;
            description "The numerical index of next-hop in
                        next-hop array";
        }

        leaf nh-weight {
            type uint32;
        }

        uses ip-next-hop;
    }
}
}

```

#### 4.4.4 YANG submodule "l2-table"

This submodule defines the per tenant L2 tables keyed by L2 MAC

addresses.

```
submodule l2-table {  
    belongs-to pse-tables {  
        prefix "pset";  
    }  
  
    import ietf-yang-types {  
        prefix "yang";  
    }  
  
    import l2-next-hop {  
        prefix "l2nh";  
    }  
  
    organization "Cisco Systems";  
    contact "joe@acme.example.com";  
  
    description  
        "The module defines a forwarding tables based on L2 MAC  
        addresses";  
  
    revision 2013-12-05 {  
        description "Initial revision.";  
    }  
    grouping l2-features {  
        uses l2nh:l2-next-hop-features;  
    }  
  
    grouping l2-table {  
        list L2-table {  
            key "dst-mac";  
  
            leaf dst-mac {  
                type yang:mac-address;  
            }  
  
            uses l2nh:l2-nh-array;  
        }  
    }  
}
```

#### 4.4.5 YANG submodule "l2-next-hop"

This submodule defines per tenant the list of different L2 next hops that can be pointed at by L2 and L3 prefixes in the L2 or L3 tables.

```
module l2-next-hop {  
  
    namespace  
    "http://www.cisco.com/something/something-else/context3";  
    prefix "l2nh";  
  
    import ietf-inet-types {  
        prefix "inet";  
    }  
  
    import pse-common-types {  
        prefix "ct";  
    }  
  
    description  
        "The module defines a set of next-hop tables  
        specific forwarding tables can be derived";  
  
    revision 2013-12-05 {  
        description "Initial revision.";  
    }  
  
    grouping l2-next-hop-features {  
        leaf l2-ecmp-supported {  
            type boolean;  
        }  
  
        leaf l2-ucmp-supported {  
            type boolean;  
        }  
  
        leaf l2-max-mp-next-hops {  
            type uint32;  
        }  
  
        container l2-nh-type-supported {  
            leaf nh-type-if {  
                type boolean;  
            }  
  
            leaf nh-type-mppls-over-gre {  
                type boolean;  
            }  
  
            leaf nh-type-vxlan {  
                type boolean;  
            }  
        }  
    }  
}
```

```
}  
  
grouping l2-next-hop {  
    choice l2-next-hop-type {  
        case l2-nh-type-if {  
            leaf l2-nh-if {  
                type ct:interfaceName;  
            }  
        }  
  
        case l2-nh-type-mpls-over-gre {  
            leaf local-gre-addr {  
                type inet:ip-address;  
            }  
  
            leaf remote-gre-addr {  
                type inet:ip-address;  
            }  
  
            leaf gre-key {  
                type uint32;  
            }  
  
            leaf mpls-label {  
                type uint32;  
            }  
        }  
  
        case l2-nh-type-vxlan {  
            leaf local-vxlan-addr {  
                type inet:ip-address;  
            }  
  
            leaf remote-vxlan-addr {  
                type inet:ip-address;  
            }  
  
            leaf vxlan-id {  
                type uint32;  
            }  
        }  
    }  
}  
  
grouping l2-nh-array {  
    list l2-nh-array {
```

```
        key next-hop-index;

        leaf next-hop-index {
            type uint8;
            description "This is an opaque index generated by the
                        device to uniquely identify a list of
                        multi-path next-hops";
        }

        leaf nh-weight {
            type uint32;
        }

        uses l2-next-hop;
    }
}
```

#### 4.4.6 YANG submodule "interface-table"

This submodule defines per tenant table the list of interfaces will be hosted on a device implementing PSE.

```
submodule interface-table {
    belongs-to pse-tables {
        prefix "pse-if";
    }
    import ietf-interfaces {
        prefix if;
    }
    import ietf-inet-types {
        prefix "inet";
    }
    import pse-common-types {
        prefix "ct";
    }
    description
        "Interface table";

    organization "Cisco Systems";
    contact "employee@cisco.com";

    revision 2013-12-05 {
        description "Initial revision.";
    }
    augment "/if:interfaces/if:interface" {
        when "if:type = 'ethernetCsmacd' or
```



```
        if:type = 'ieee8023adLag';
    leaf vlan-tagging {
        type boolean;
        default false;
    }
}
augment "/if:interfaces/if:interface" {
    when "if:type = 'l2vlan'";
    leaf base-interface {
        type if:interface-ref;
        must "/if:interfaces/if:interface[if:name = current()]"
            + "/if:vlan-tagging = 'true'" {
            description
                "The base interface must have vlan tagging enabled.";
        }
    }
    leaf outer-vlan-id {
        type uint16 {
            range "1..4094";
        }
        must "../base-interface" {
            description
                "If a vlan-id is defined, a base-interface must
                be specified.";
        }
    }
    leaf inner-vlan-id {
        type uint16 {
            range "1..4094";
        }
        must "../base-interface" {
            description
                "If a vlan-id is defined, a base-interface must
                be specified.";
        }
    }
}
augment "/if:interfaces/if:interface" {
    when "if:type = 'ethernetCsmacd'";

    container ethernet {
        must "../if:location" {
            description
                "An ethernet interface must specify the physical
                location of the ethernet hardware.";
        }
        choice transmission-params {
            case auto {
```

```

        leaf auto-negotiate {
            type empty;
        }
    }
    case manual {
        leaf duplex {
            type enumeration {
                enum "half";
                enum "full";
            }
        }
        leaf speed {
            type enumeration {
                enum "10Mb";
                enum "100Mb";
                enum "1Gb";
                enum "10Gb";
            }
        }
    }
}

augment "/if:interfaces/if:interface" {
    leaf ip-enabled {
        type boolean;
        default false;
    }
}

augment "/if:interfaces/if:interface" {
    leaf base-ip {
        type if:interface-ref;
        must "/if:interfaces/if:interface[if:name = current()]"
            + "/if:ip-enabled = 'true'" {
            description
                "The base interface must have vlan tagging enabled.";
        }
    }
    leaf ipv4-addr {
        type inet:ipv4-address;
        must "../base-interface" {
            description
                "If an ip address is defined, a base-interface must
                be specified.";
        }
    }
    leaf ipv4-prefix-length {
        type ct:prefixLengthIPv4;
    }
}

```

```

        must "../base-interface" {
            description
                "IPv4 address prefix length";
        }
    }
    leaf ipv6-addr {
        type inet:ipv6-address;
        must "../base-interface" {
            description
                "If an ip address is defined, a base-interface must
                be specified.";
        }
    }
    leaf ipv6-prefix-length {
        type ct:prefixLengthIPv6;
        must "../base-interface" {
            description
                "IPv6 address prefix length";
        }
    }
    leaf mtu {
        type uint32;
        description
            "The size, in octets, of the largest packet that the
            interface can send and receive. This node might not be
            valid for all interface types.

            Media-specific modules must specify any restrictions on
            the mtu for their interface type.";
    }
    leaf unnumbered-to-intf {
        type if:interface-ref;
    }
}
grouping if-table {
    leaf table-int {
        type if:interface-ref;
    }
}
}
}

```

#### 4.4.7 YANG submodule "arp-table"

This submodule defines per tenant the list of ARP entries that will be hosted on a device implementing PSE.

```

submodule arp-table {
    belongs-to pse-tables {

```

```
        prefix "arp";
    }

    import ietf-inet-types {
        prefix "inet";
    }

    import ietf-yang-types {
        prefix "yang";
    }

    import pse-common-types {
        prefix "ct";
    }

    organization "Cisco Systems";
    contact "joe@acme.example.com";

    description
        "The module defines a forwarding tables based on IPv4/V6
        addresses";

    revision 2013-12-05 {
        description "Initial revision.";
    }

    grouping arp-table {
        list arptable {
            key "ip-address";
            leaf ip-address {
                type inet:ip-address;
            }

            leaf mac-address {
                type yang:mac-address;
                description "Destination MAC Address";
            }

            leaf interface-name {
                type ct:interfaceName;
            }
        }
    }
}
```

#### 4.4.8 Yang submodule "arp-proxy-table"

This submodule defines per tenant proxy-arp entries that will be

hosted on a device implementing PSE.

```
submodule arp-proxy-table {
  belongs-to pse-tables {
    prefix "arp-proxy";
  }

  import ietf-inet-types {
    prefix "inet";
  }

  organization "Cisco Systems";
  contact "joe@acme.example.com";

  description
    "The module defines ARP Proxy tables based on IPv4/V6
    addresses";

  revision 2013-12-05 {
    description "Initial revision.";
  }

  grouping arp-proxy-table {
    list arpproxytable {
      key "ip-prefix";
      leaf ip-prefix {
        type inet:ip-prefix;
      }
      leaf gateway-ip-address {
        type inet:ip-address;
      }
    }
  }
}
```

## 5 Performance and scale requirements

Encoding/decoding of forwarding updates must be extremely efficient, to achieve high performance in terms of number of forwarding updates/second from control plane to forwarding plane. The transport protocol must allow for negotiation for different encoding methods.

To achieve high convergence rate of updates, async updates MUST be used and this can happen by relaxing the dependencies and constraints between the different modules and objects in the model, leaving the PSE to resolve the routes forwarding chains via the nexthop and the different encaps.

The details of an efficient encoder/decoder is outside the scope of this document.

## 6 YANG data model tree

```

module: pse-tables
  +--ro pse-global-features
  |   +--ro ready?                boolean
  |   +--ro ip-unicast-forwarding? boolean
  |   +--ro l2-forwarding?        boolean
  |   +--ro flow-forwarding?      boolean
  |   +--ro ipv4-unicast-features
  |   |   +--ro ipv4-features
  |   |   |   +--ro ecmp-supported?    boolean
  |   |   |   +--ro ucmp-supported?    boolean
  |   |   |   +--ro max-mp-next-hops?  uint32
  |   |   |   +--ro nh-type-supported
  |   |   |   |   +--ro nh-type-regular?    boolean
  |   |   |   |   +--ro nh-type-mppls-over-gre? boolean
  |   |   |   |   +--ro nh-type-vxlan?      boolean
  |   +--ro ipv6-unicast-features
  |   |   +--ro ipv6-features
  |   |   |   +--ro ecmp-supported?    boolean
  |   |   |   +--ro ucmp-supported?    boolean
  |   |   |   +--ro max-mp-next-hops?  uint32
  |   |   |   +--ro nh-type-supported
  |   |   |   |   +--ro nh-type-regular?    boolean
  |   |   |   |   +--ro nh-type-mppls-over-gre? boolean
  |   |   |   |   +--ro nh-type-vxlan?      boolean
  |   +--ro l2-ecmp-supported?    boolean
  |   +--ro l2-ucmp-supported?    boolean
  |   +--ro l2-max-mp-next-hops?  uint32
  |   +--ro l2-nh-type-supported
  |   |   +--ro nh-type-if?        boolean
  |   |   +--ro nh-type-mppls-over-gre? boolean
  |   |   +--ro nh-type-vxlan?      boolean
  |   +--ro ecmp-supported?        boolean
  |   +--ro ucmp-supported?        boolean
  |   +--ro max-mp-next-hops?      uint32
  |   +--ro nh-type-supported
  |   |   +--ro nh-type-regular?    boolean
  |   |   +--ro nh-type-mppls-over-gre? boolean
  |   |   +--ro nh-type-vxlan?      boolean
  +--rw pse-oam
  |   +--rw prefix-table
  |   |   +--rw unicast-prefix [prefix]
  |   |   |   +--rw prefix      inet:ip-prefix

```

```

+--rw interface-table
|   +--rw interfaces [interface-name]
|       +--rw interface-name      ct:interfaceName
+--ro oper-prefix-down-table
|   +--ro oper-prefix-down-list [version]
|       +--ro version      uint32
|       +--ro prefix?      inet:ip-prefix
+--ro oper-interface-down-table
|   +--ro oper-interfaces-down-list [version]
|       +--ro version      uint32
|       +--ro interface-name? ct:interfaceName
+--rw sync
|   +--rw sync-state?  enumeration
+--rw context-selector-table
|   +--rw label-ctx-selection-table
|       +--rw label-entry [label]
|           +--rw label      uint32
|           +--rw disposition-table
|               +--rw pse-ctx-name?  string
|               +--rw ctx-tbl-name?  string
+--rw interface-ctx-selector-table
|   +--rw interfaces [interface-name selection-type]
|       +--rw interface-name      ct:interfaceName
|       +--rw selection-type      tbl-selection-type
|       +--rw (tbl-selection-type)?
|           +--:(ipv4)
|               +--rw ipv4-table
|                   +--rw pse-ctx-name?  string
|                   +--rw ctx-tbl-name?  string
|           +--:(ipv6)
|               +--rw ipv6-table
|                   +--rw pse-ctx-name?  string
|                   +--rw ctx-tbl-name?  string
+--rw label-table [local-label]
|   +--rw local-label      uint32
+--rw label-attributes
|   +--rw local-label-type?  LocalLabelType
|   +--rw (local-label-type-choice)?
|       +--:(head-prefix)
|           +--rw prefix?      inet:ip-prefix
|           +--rw pref-table
|               +--rw pse-ctx-name?  string
|               +--rw ctx-tbl-name?  string
+--rw mpls-nh-array [next-hop-index]
|   +--rw next-hop-index      uint8
|   +--rw nh-weight?          uint32
|   +--rw (ip-next-hop-type)?
|       +--:(ip-nh-type)

```

[ Page 44 ]



```

+--rw tables [table-name]
+--rw table-name          string
+--rw pse-vpn-id?         string
+--rw (pse-table-type)?
+--:(ip-unicast-table)
+--rw arp-table
+--rw arptable [ip-address]
+--rw ip-address          inet:ip-address
+--rw mac-address?       yang:mac-address
+--rw interface-name?    ct:interfaceName
+--rw arp-proxy-table
+--rw arpproxytable [ip-prefix]
+--rw ip-prefix           inet:ip-prefix
+--rw gateway-ip-address? inet:ip-address
+--rw ip-unicast-table
+--rw ip-tbl-attrs
+--rw afi?               ianaaf:address-family
+--rw unicast-prefix [prefix]
+--rw prefix             inet:ip-prefix
+--rw ip-nh-array [next-hop-index]
+--rw next-hop-index     uint8
+--rw nh-weight?        uint32
+--rw (ip-next-hop-type)?
+--:(ip-nh-type)
+--rw ip-next-hop?      inet:ip-address
+--rw interface?       ct:interfaceName
+--rw nh-tbl-name
+--rw pse-ctx-name?     string
+--rw ctx-tbl-name?    string
+--:(ip-nh-type-mpls-gre)
+--rw local-gre-addr?   inet:ip-address
+--rw remote-gre-addr? inet:ip-address
+--rw nh-gre-tbl-name
+--rw pse-ctx-name?     string
+--rw ctx-tbl-name?    string
+--rw mpls-label?      uint32
+--rw gre-key?         uint32
+--:(ip-nh-type-vxlan)
+--rw local-vxlan-addr? inet:ip-address
+--rw remote-vxlan-addr?inet:ip-address
+--rw vxlan-id?        uint32
+--ro per-prefix-stats
+--ro packets?         yang:counter64
+--ro bytes?          yang:counter64
+--:(l2-table)
+--rw l2-table
+--rw L2-table [dst-mac]
+--rw dst-mac          yang:mac-address

```

```

|      +--rw l2-nh-array [next-hop-index]
|      +--rw next-hop-index      uint8
|      +--rw nh-weight?          uint32
|      +--rw (l2-next-hop-type)?
|      |      +--:(l2-nh-type-if)
|      |      |      +--rw l2-nh-if?      ct:interfaceName
|      |      +--:(l2-nh-type-mppls-over-gre)
|      |      |      +--rw local-gre-addr? inet:ip-address
|      |      |      +--rw remote-gre-addr? inet:ip-address
|      |      |      +--rw gre-key?      uint32
|      |      |      +--rw mppls-label?   uint32
|      |      +--:(l2-nh-type-vxlan)
|      |      |      +--rw local-vxlan-addr? inet:ip-address
|      |      |      +--rw remote-vxlan-addr? inet:ip-address
|      |      |      +--rw vxlan-id?      uint32
+--:(flow-table)
+--rw flow-table
+--rw afi?      ianaaf:address-family
+--rw flow-table
+--rw src-address      inet:ip-address
+--rw dst-address      inet:ip-address
+--rw src-port         uint32
+--rw dst-port         uint32
+--rw protocol-type    string
+--rw nh-array
+--rw ip-nh-array [next-hop-index]
+--rw next-hop-index      uint8
+--rw nh-weight?          uint32
+--rw (ip-next-hop-type)?
+--:(ip-nh-type)
|      +--rw ip-next-hop?  inet:ip-address
|      +--rw interface?    ct:interfaceName
|      +--rw nh-tbl-name
|      |      +--rw pse-ctx-name?  string
|      |      +--rw ctx-tbl-name?  string
+--:(ip-nh-type-mppls-gre)
|      +--rw local-gre-addr?inet:ip-address
|      +--rw remote-gre-addr? inet:ip-address
|      +--rw nh-gre-tbl-name
|      |      +--rw pse-ctx-name?  string
|      |      +--rw ctx-tbl-name?  string
|      +--rw mppls-label?      uint32
|      +--rw gre-key?          uint32
+--:(ip-nh-type-vxlan)
+--rw local-vxlan-addr? inet:ip-address
+--rw remote-vxlan-addr? inet:ip-address
+--rw vxlan-id?      uint32

```

## 7 Major Contributing Authors

The editors would like to thank Reshad Rahman, Yuri Tsier, and Kausik Majumdar who made a major contribution to the development of this document.

Reshad Rahman  
Cisco  
Email: rrahman@cisco.com

Yuri Tsier  
Cisco  
Email: ytsier@cisco.com

Kausik Majumdar  
Cisco  
Email: kmajumda@cisco.com

## 8 Security Considerations

This document does not introduce any additional security constraints.

## 9 IANA Considerations

TBD

## 10 References

### 10.1 Normative References

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

### 10.2 Informative References

## Authors' Addresses

Rex Fernando  
Cisco  
Email: rex@cisco.com

Sami Boutros  
Cisco  
Email: sboutros@cisco.com

Dhananjaya Rao  
Cisco  
Email: dhrao@cisco.com

Nabil Bitar  
Verizon  
Email:nabil.n.bitar@verizon.com

Luay Jalil  
Verizon  
Email:luay.jalil@verizon.com

I2RS  
Internet-Draft  
Intended status: Standards Track  
Expires: November 9, 2014

T. Petch  
Engineering Networks Ltd  
May 8, 2014

Interactions of I2RS with NETCONF and YANG  
draft-tp-i2rs-yang-00.txt

Abstract

This memo looks at some possible interactions between the requirements of I2RS and the current capabilities (and data models) of YANG and NETCONF.

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 November 9, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	2
2. Operational State . . . . .	2
3. Instantiation . . . . .	3
4. Defaults . . . . .	4
5. Editable state . . . . .	5
6. Persistence . . . . .	6
7. Stability . . . . .	8
8. Security Considerations . . . . .	8
9. Acknowledgements . . . . .	8
10. References . . . . .	8
10.1. Normative References . . . . .	8
10.2. Informative References . . . . .	9
Author's Address . . . . .	9

## 1. Introduction

This memo looks at some possible interactions between the requirements of I2RS and the current capabilities (and data models) of YANG [RFC6020] and NETCONF [RFC6241].

## 2. Operational State

YANG and NETCONF characterise data as being one of two types, config and state. State may then be further subdivided into operational state and read-only statistics. This terminology has been around since at least RFC3535 (2002) when the operators told the IAB of some limitations of SNMP. The sort of feature that was missing was the ability to pull the config from a box, replace the box, push the config back to a replacement box which would then resume operation, as if nothing had changed (give or take a few flaps and Graceless Restarts of the routing protocols). Crucially, config, in this context, is the changes made from the default state by CLI (or some such) to enable the box to operate. The rest of the data in a box is state, which is then divided into read-only statistics (the only kind thereof) and - Operational State, which is everything else that the box has acquired, be it from BGP or OSPF, from DHCP or NTP or ND, or parameters set up by the box itself, for example in response to detecting the insertion of a new blade in a card slot. (Such data is sometimes referred to as System Controlled).

Thus entries in a routing table learnt from BGP are Operational State; a static entry put in by CLI (or some such) is config. The parameters for a hot-plugged interface card are Operational State; turning on passive ospf on that interface is config.

YANG (along with NETCONF) is designed for writing configuration, and reading everything else. Configuration means, in the YANG context, what you might put in through the CLI and it excludes everything else, such as data learnt via a protocol or created by the system.

YANG defines data nodes with the container, leaf, leaf-list, list and anyxml statements, each of which can take the YANG substatement 'config', which can take the values true or false. 'config false' is then state, is read only and is read with a NETCONF get RPC. 'config true' is config, can be written and can be read with a NETCONF get-config RPC.

NETCONF contains powerful filtering capabilities and these filters could have been used to extract 'config true' data from a box but the resulting RPC would be complex and error-prone; the NETCONF get-config RPC allows simple retrieval of the data of greatest interest to the users of NETCONF, i.e. configuration data. Thus 'config true' does not just make it possible to edit the data node - it also defines a subset of the data model that can be retrieved with a single, get-config, RPC. The set of data that is editable and the set that should be retrieved as the configuration is one and the same; this coincidence of the two sets, happily, appears not to have generated any issues.

The value of the config substatement cascades down to subordinate nodes - YANG models are currently a tree structure - so that in practice, the statement need only be specified in a few places at the highest level of a tree.

### 3. Instantiation

One restriction is that while 'config false' nodes may appear in the subtree of 'config true' nodes, 'config true' nodes may not appear in the subtree of 'config false' nodes. One quirk arising from this, that may not be immediately apparent, is that 'config false' nodes in the subtree of a 'config true' node do not get instantiated in the data model unless and until the 'config true' node is configured. Thus operational state data may be inaccessible unless and until some configuration is performed. This would apply to tables, such as the interface table, the routing table or a prefix list for an interface. The treatment of this issue has evolved and is still evolving.

It led, initially, to a twin trees approach in defining data models, as can be seen in the current system, interfaces-cfg and routing-cfg YANG data models, with one read-write tree of configured information and a separate read-only tree of state information. (draft-ietf-netmod-interfaces-cfg-10 is an example of the earlier approach). In the case of a routing table, the configured table may be more or less

empty with all the information coming via routing protocols, and so being (operational) state. In the case of interfaces, then there could be more or less complete duplication, with every interface appearing in both lists. If a user of NETCONF wished to obtain all the relevant data, it would then have been necessary to obtain data from both trees.

YANG provides no way to correlate entries in such a pair of lists, such as of routes or interfaces. Rather, the description of the data model must specify how entries should be matched between one list and another. Thus the routing model, in [www.ietf.org/id/draft-ietf-netmod-routing-cfg-13.txt](http://www.ietf.org/id/draft-ietf-netmod-routing-cfg-13.txt) specifies that

"Corresponding entries in both versions of the list (in operational state data and configuration) have the same value of the list key."

An evolution of this position is that entries in a configuration list are automatically copied into the operational state list so that now, only the operational state list need to read in order to obtain a comprehensive picture. Again, this is something that must be specified in the description of the data model; YANG provides no assistance in this regard. Again, quoting from the routing model,

"If the server accepts a configured user-controlled entry, then this entry also appears in the operational state version of the list."  
"

where user-controlled refers to data input as config via NETCONF.

As data models evolve, it seems possible, if not even likely, that the paralleling of data between (operational) state and config will be the norm and not just restricted to a few lists - there seems little that can only ever appear as config, perhaps such statements as 'ospf passive' or 'bgp 192.168.1.1 as 65533' (important as such statements are).

#### 4. Defaults

While the split between config and state, the latter being any 'config false' node, has been well-defined since the early days of NETCONF, the question of whether or not this two-way split is sufficient has led to much discussion, with a proposal for up to three boolean substatements for YANG data nodes, leading to eight possible combinations with much discussion as to which might be valid or useful.



One feature to emerge from this is the handling of defaults. Much of what might be configured can be left to take a default value as specified in the data model, so that the task of configuring a box can be simplified if default values may be omitted. On the other hand, there are times when the existence of a default may be the key to understanding why a system is performing as it is and so it should also be possible to read default values that the system is using. Then there are times when a data node is configured with the value that is the default value and this may, or may not, be regarded as being in the same condition as if the value were not configured at all.

Three possible treatments of default values were identified, all regarded as valid, and all are specified in [RFC6243]. Note that while YANG can specify the existence of a default value, the treatment of it, as described above, is part of NETCONF - YANG did not acquire any additional substatements to defined the treatment of defaults.

## 5. Editable state

By contrast, the issue of whether or not it is possible (or desirable) to be able to edit operational state has continued to generate discussion and here YANG has a part to play. Clearly such editing is a requirement for I2RS. YANG is an extensible language so an additional substatement can be added to data nodes, such as

```
i2rs:edit-data
```

where edit-data might be a boolean taking the values true or false. (The i2rs: is a prefix that is linked to the namespace in which the substatement, or indeed any element of YANG that is not in the original definition thereof, resides). This substatement then overrides one of 'config false' in that it makes the node editable.

This substatement could be modelled on the config substatement, in that it applies to all nodes in the subtree unless overridden. The substatement would, presumably, only be valid on 'config false' nodes.

Earlier, it was pointed out that 'config true' served a dual purpose; it defines the set of nodes that can be edited and it defines the set that it is useful to retrieve, with a NETCONF get-config RPC.

It seems likely, to the author, that this logic does not apply to I2RS operations on operational state, that is the set of data that I2RS may want to edit will be a subset of the data that is of interest to I2RS; read-only statistics, for example, will not be in

the former but will be in the latter. From this it follows that while a single, boolean substatement may be sufficient, and that anything else can be achieved with filters, it would much simpler and less error-prone for I2RS to have a further substatement that defines the set of data that is of interest to I2RS so that such data can be readily retrieved by an additional NETCONF RPC, `get-i2rs` perhaps (NETCONF, like YANG, is extensible).

Such substatements would, like `config`, apply to all nodes in the subtree unless overridden. By analogy with the `config` substatement, it would seem likely that an `'edit-data true'` node should not appear under an `'edit-data false'` node (although quite why is not clear to the author).

## 6. Persistence

YANG has no concept of persistence and only a limited one of a datastore. YANG defines a schema tree; whether there is one or more than one instantiation thereof and where they are is outside the scope of YANG, but is very much a part of NETCONF.

NETCONF defines a configuration datastore as the data that is required to get a device from its initial default state into a desired operational state (presuming that protocols such as BGP and DHCP will then do the rest of the work that is needed). By default, NETCONF defines a single, running configuration datastore to which NETCONF operations are applied. Curiously, the persistence thereof appears not to be specified, the assumption appearing to be that changes made thereto are persistent and will be present after a reboot of a box.

Optionally, there can be other datastores, such as a startup configuration datastore. Now changes made to the running datastore are not persistent unless and until a NETCONF `copy-config` RPC is performed, from running to startup.

NETCONF defines a candidate configuration datastore while users can define any number of additional datastores.

A datastore is not just a coherent collection of configuration data. YANG contains powerful logical capabilities that allow the validity checking of a configuration, with statements such as

```
when "../type = 'read' or ../type = 'write'";
```

or

```
augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route" {  
  when "../..//rt:address-family = 'v6ur:ipv6-unicast'" { } ... }
```

The question for I2RS is whether or not the concept of a datastore is applicable, for either of the reasons given above.

Will I2RS, having made changes to the operational state of a box, want to copy those changes to somewhere else, perhaps so that they can be re-applied at a later time, such as after a reboot, or is the underlying assumption of I2RS that such changes should be lost at a reboot? Will such a copy be required as an audit trail, a check on what actually happened, given that the changes that I2RS make are likely to be of higher risk to the integrity of the routing system than those made via configuration and NETCONF?

If copying is required, where will the copies be to? on the box or, as seems more likely, back to the rest of the I2RS infrastructure, the I2RS clients? Will I2RS also want to create a set of changes on the box, like a candidate configuration datastore, so that they can be reviewed, validity checked by YANG logical statements, before they are applied?

If such copy operations are required, then how will the data that is to be copied be identified? NETCONF deals with a relatively simple set of data, the configuration data that is needed to take a box to its operational state. (Even then, the picture is complicated by issues such as defaults). With I2RS, the amount of data involved is, potentially much greater - e.g. the full BGP table - if not so diverse - NETCONF must deal with any aspect of configuration whereas I2RS, at least initially, is focussed on routing. Will I2RS need an additional YANG substatement to specify which nodes should form part of an I2RS datastore? Will there need to be different types of datastore for different types of data within I2RS? As before, the current concept of configuration makes these issues relatively simple for NETCONF, more complex where I2RS is concerned.

Equally it seems unclear whether or not the other property of datastores, that of being able to specify validity checks to be performed on the data, is applicable to I2RS or not, both in the sense of whether or not it makes sense for I2RS to have that functionality and whether or not the YANG statements make sense when applied to I2RS state data. Currently such checks are defined as valid for any configuration datastore, or for all state data on the device along with the running datastore, the latter being applicable to the results of a NETCONF get RPC. The introduction of editable operational state complicates this picture further.

The answers to these questions may lie with the uses to which I2RS is put; tightly scoped use cases would be a help here.

## 7. Stability

Finally, one unanswered question (where use cases would again be of help) is what data it is that I2RS is likely to want to change, within, say, the routing system. A routing system is stable in some sense of that word (arguably, there will always be some change in the global Internet that has yet to propagate all the way to all routers but within a limited radius of the box in question, the routing system may be considered stable).

A stable system in engineering is one that, when perturbed, returns to its original state. So make a change in one router and the rest of the routing system may seek to update that router and reinstate the status quo. For example, at least conceptually, a change to BGP's Adj-RIB-Out will be refreshed from Adj-RIB-In and so lost, while a change to Adj-RIB-In will be lost when the router that provided the data repeats its advertisement of the route.

Thinking of the changes I2RS might want to make, it seems likely that many, or most, will be changes to what NETCONF/YANG refer to as config and not to the operational state. Whether such changes are ephemeral or persistent, in the long term, is then a question of which NETCONF configuration datastore the changes are made to, running or startup or both.

Again, use cases might clarify this.

## 8. Security Considerations

This informational memo introduces no security considerations.

## 9. Acknowledgements

## 10. References

### 10.1. Normative References

- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

## 10.2. Informative References

[RFC6243] Bierman, A. and B. Lengyel, "With-defaults Capability for NETCONF", RFC 6243, June 2011.

## Author's Address

Tom Petch  
Engineering Networks Ltd  
18 Parkwood Close  
Lymm, Cheshire WA13 0NQ  
UK

Email: [tomSecurity@network-engineer.co.uk](mailto:tomSecurity@network-engineer.co.uk)