

I2RS working group
Internet-Draft
Intended status: Informational
Expires: May 19, 2017

S. Hares
Huawei
A. Dass
Ericsson
November 15, 2016

Yang for I2RS Protocol
draft-hares-netmod-i2rs-yang-01.txt

Abstract

This document requests one yang model addition that will support ephemeral state and provides notes for the implementers who wish to implement ephemeral state for the I2RS Protocol. The purpose of this document is to provide implementers of ephemeral state with background and open issues that they should consider when implementing ephemeral state that satisfies the I2RS protocol.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 19, 2017.

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. Definitions Related to Ephemeral Configuration	3
2.1. Requirements language	3
2.2. I2RS Definitions	3
3. Overview of Changes	6
3.1. I2RS protocol requirements	6
3.2. NETCONF Features and Extensions	6
3.3. RESTCONF features and Extensions	7
3.4. Assumptions on Data Store Model Melee	8
4. Ephemeral Data	8
4.1. Ephemeral Control Plane Datastore	9
4.2. Qualities of Ephemeral Datastore	10
4.3. I2RS Agent Caching of Ephemeral Data	10
4.4. Massive Amounts of Configuration Data	11
4.5. Write Error handling	12
4.5.1. Normal validation checks	12
4.6. IPFIX for traffic monitoring	12
4.7. Binary encoding of RESTCONF/NETCONF	13
4.8. Ephemeral state in DDoS environments	13
5. Yang Changes	13
6. IANA Considerations	14
7. Security Considerations	14
8. Acknowledgements	14
9. References	15
9.1. Normative References:	15
9.2. Informative References	17
Authors' Addresses	19

1. Introduction

This a proposal for yang additions to support the first version of the I2RS protocol.

The I2RS architecture [RFC7921] defines the I2RS interface "a programmatic interface for state transfer in and out of the Internet routing system". The I2RS protocol is a protocol designed to a higher level protocol comprised of a set of existing protocols which have been extended to work together to support a new interface to the routing system. The I2RS protocol is a "reuse" management protocol which creates new management protocols by reusing existing protocols and extending these protocols for new uses, and has been designed to be implemented in phases [RFC7921].

The first version of the I2RS protocol is comprised of extensions to existing features of NETCONF [RFC6241] and RESTCONF [I-D.ietf-netconf-restconf]. The data modeling language for the I2RS protocol will be Yang [RFC7950] with features and extensions proposed in this draft.

The structure of this document is:

Section 2 provides definitions for terms in this document.

Section 3 summarizes the changes to configuration data store, NETCONF, RESTCONF, and YANG.

[I-D.ietf-i2rs-ephemeral-state] specifies the I2RS requirements for the ephemeral state. Section 4 discusses how these requirements might be implemented in a control plane datastore.

Section 5 describes the one required Yang model addition for I2RS (ephemeral key word). This section also describes elements of information in the NETCONF/RESTCONF implementations that must be queryable by the I2RS protocol implementations.

2. Definitions Related to Ephemeral Configuration

This section reviews definitions from I2RS architecture [RFC7921] and NETCONF operational state definitions [I-D.nmdsdt-netmod-revised-datastores] before using these to construct a definition of the ephemeral data store.

2.1. Requirements language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2.2. I2RS Definitions

The I2RS architecture [RFC7921] defines the following terms:

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. (See [RFC7921] for an architectural overview, [I-D.ietf-i2rs-ephemeral-state] for requirements, and [I-D.nmdsdt-netmod-revised-datastores] for discussion of how the ephemeral datastore as a control plane datastore interacts with

intended datastore and dynamic configuration protocols to form the applied datastore".

local configuration: is the data on a routing system which does persist across a reboot (software or hardware) and a power on/off condition. Local configuration has the ability to roll back to a pervious configuration state. Local configuration is defined as the intended datastore [I-D.nmdsdt-netmod-revised-datastores] which is modified by dynamic configuration protocols (such as DHCP) and the I2RS ephemeral data store

dynamic configuration protocols datastore are configuration protocols such as DHCP that interact with the intended datastore (which does persist across a reboot (software or hardware) power on/off condition), and the I2RS ephemeral state control plane datastore.

operator-applied policy: is a policy that an operator sets that determines how the ephemeral datastore as a control plane data store interacts with applied datastore (as defined in [I-D.nmdsdt-netmod-revised-datastores]). This operator policy consists of policy knobs that the operator sets to determine how the I2RS agent control plane ephemeral state datastore will interact with the intended configuration datastor and the dynamic configuration protocol datastore. Three policy knobs could be used to implement this policy:

- * policy knob 1: I2RS Ephemeral control-plane datastore takes takes precedence over the intended datastore in the routing protocols.
- * policy knob 2: Updated intended configuration datastore takes precedence over the I2RS ephemeral control-plane data store in the routing protocols
- * policy knob 3: Ephemeral control plane datastore takes precedence over any other dynamic configuration protocols datastore.

An practical example for three states of the operator-applied policy may help the reader understand the concept. Consider the following three desired outcomes with their policy knob states:

Monitoring Features only The policy knob settings are:

Policy knob 1=false,
policy knob 2=true,

Policy knob 3=false,

Action: I2RS protocol software feature is installed, but the operator does not want the I2RS ephemeral datastore to take precedence (that is be used) on any variables in the applied configuration datastore. This policy set might be valid if I2RS is only suppose to monitor data on this node through newly defined parameters.

I2RS Agent Changes win the policy knob settings would be:

Policy knob 1=true,

policy knob 2=false,

Policy knob 3=false,

Action: This is the normal case for the I2RS Agent where the ephemeral control-plane datastore takes precedence over the intended configuration datastore and dynamic configuration datastores. The values from the I2RS ephemeral datastore are used rather than the intended configuration datastore and the dynamic configuration protocol datastore. When the ephemeral data is removed by the I2RS agent, the dyanmic configuration datastore and the intended configuration datastore state is restored, combined and passed to the routing protocols for application.

Just change until next configuration update the policy knob settings would be:

Policy knob 1=true,

policy knob 2=true,

Policy knob 3=false,

Action: This case can occur if the I2RS Client write to the ephemeral control plane data store is only suppose to take precedence until the next configuration cycle from a centralized system. Suppose the local configuration is get by the centralized system at 11:00pm each night. The I2RS Client writes temporary changes to the routing system via the I2RS agent ephemeral write. At 11:00pm, the local configuration update overwrite the ephemeral. The I2RS Agent notifies the I2RS Client which is tracking which of the ephemeral changes are being overwritten.

3. Overview of Changes

This overview reviews the following:

- o What NETCONF [RFC6241] protocol existing features required for I2RS protocol and what extension for these extension features that are needed for the I2RS protocol version 1,
- o What RESTCONF [I-D.ietf-netconf-restconf] protocol existing features are required for the I2RS protocol and what extensions are needed for I2RS protocol version 1.
- o An overview of the Yang 1.1 data modeling language[RFC7950] features are needed for I2RS protocol version 1.
- o An overview of the extensions to Yang 1.1 data modeling language [RFC7950] that are needed for the I2RS protocol version 1.

3.1. I2RS protocol requirements

The requirements for the I2RS protocol are defined in the following documents:

- o I2RS Problem Statement [RFC7920],
- o I2RS Architecture [RFC7921],
- o I2RS Traceability [RFC7922],
- o Publication and Subscription [RFC7923],
- o I2RS Ephemeral State Requirements, ,
[I-D.ietf-i2rs-ephemeral-state]
- o I2RS Protocol Security Requirements,
[I-D.ietf-i2rs-protocol-security-requirements]

The Interface to the routing System (I2RS) creates a new capability for the routing systems, and with greater capabilities come a greater need for security. The requirements for a secure environment for I2RS is described in [I-D.ietf-i2rs-security-environment-reqs].

3.2. NETCONF Features and Extensions

The features the I2RS protocol requires are:

- o NETCONF [RFC6241] with its updates [RFC7803],

- o Network Access Control Model [RFC6536] with update (draft-bierman-netconf-rf6536bis)
- o Running NETCONF over TLS with mutually X.509 authentication [RFC7589]
- o Keystore Model [I-D.ietf-netconf-keystore],
- o Subscribing to Yang Datastore updates [I-D.ietf-netconf-yang-push],
- o NETCONF support for Event Notifications [I-D.ietf-netconf-netconf-event-notifications],
- o Subscribing to NETCONF Events (updated) [I-D.ietf-netconf-rfc5277bis]
- o Yang Patch Media type [I-D.ietf-netconf-yang-patch],
- o NETCONF/RESTCONF Zero Touch provisioning [I-D.ietf-netconf-zerotouch],
- o TLS Client and Server Models [I-D.ietf-netconf-restconf-client-server]
- o Call Home [I-D.ietf-netconf-call-home],
- o Module library [RFC7895],
- o NETCONF/RESTCONF Zero Touch provisioning [I-D.ietf-netconf-zerotouch],

3.3. RESTCONF features and Extensions

This protocol strawman utilizes the following existing proposed features for NETCONF and RESTCONF

- o RESTCONF [I-D.ietf-netconf-restconf]
- o Module library [RFC7895],
- o Publication/Subscription via Push [I-D.ietf-netconf-yang-push],
- o Patch [I-D.ietf-netconf-yang-patch],
- o syslog yang module (both [RFC5424] and [I-D.ietf-netmod-syslog-model])

3.4. Assumptions on Data Store Model Melee

The NETMOD Working Group has been working to create new definitions of datastores based on feedback from operators on desiring a split between operational state and configuration state.

This document takes [I-D.nmdsdt-netmod-revised-datastores] as the current status of the datastore discussion on configuration state, operational state, ephemeral state changes (via I2RS), and routing protocol state. The following things need to be carefully defined in this work:

- What is a dynamic configuration protocol (is it I2RS or DHCP)

- What is a control-plane datastore - (ephemeral state only or others?)

- How to express the policy knobs that provide preference between intended configuration, control plane datstore, and dynamic configuration protocols

- How does operational state allow for operational state to be defined by ephemeral-only data models, and mixed (ephemeral + intended configuration)

[I-D.nmdsdt-netmod-revised-datastores] is making good progress, but these additional details need to be tied down.

4. Ephemeral Data

This section provides an overview of the ephemeral data store as a control plane datastore and discusses several concepts that implementers need to consider and provide feedback on. The concepts include basic ephemeral datastore concepts, I2RS caching of ephemeral data, issues for massive data flow, error handling (normal and reduced), use of IPFIX or Binary for carrying I2RS ephemeral data, and ephemeral state.

This section augments [I-D.nmdsdt-netmod-revised-datastores] to begin to discuss how the ephemeral state control-plane datastore might be implemented.

The purpose of this section is to gather implementer wisdom on the ephemeral datastore into one place. This section discusses:

- Ephemeral state as a control plane data store

- Qualities of ephemeral datastores

Need to support Massive amounts of configuration data,

Two types of Error handling (regular, reduced)

Should we support link to IPFIX in I2RS protocol and ephemeral state?

Binary encoding for RESTCONF/NETCONF

Ephemeral state in DDoS environments.

[I-D.ietf-i2rs-ephemeral-state] describes the requirements for I2RS ephemeral state.

This section augments [I-D.nmdsdt-netmod-revised-datastores] to begin to discuss how the ephemeral state control-plane datastore might be implemented. This initial draft refines the general description so that early I2RS ephemeral state implementations may progress.

4.1. Ephemeral Control Plane Datastore

[I-D.nmdsdt-netmod-revised-datastores] architecture suggests that the applied configuration is the combination of intended datastore, the dynamic configuration protocols, and the control-plane datastores. As described above, there are policy knobs which allow the I2RS Agent to handle deciding what specific configuration variables is installed in protocols (E.g BGP) or protocol independent functions (RIB or Filters). In addition, the control-plane datastore may store the parameters need to provide publication of events, statistics, telemetry within the ephemeral control-plane datastore.

The ephemeral data-store may have models which learn operational state and augment it by configuration. For example [I-D.ietf-i2rs-yang-l3-topology] uploads ospf and isis topology information from the routing system and allows configuration of additional links or nodes.

This new architecture is a multiple panes-of-glass model where the decision on what value is chosen is based on policy. The extension of this model is that it is possible for two or more of the control-plane datastores to be ephemeral. If this occurs, then the policy knobs must define the how the 2+ ephemeral datastores interact with each other and the configuration state.

4.2. Qualities of Ephemeral Datastore

Note: The requirements for ephemeral state are in:
[I-D.ietf-i2rs-ephemeral-state]).

This section provides a discussion so that implementers writing code for these datastores can discuss what needs to be standardized and what does not need to be standardized.

The ephemeral data store has the following general qualities:

1. Ephemeral state is not unique to I2RS work.
2. The ephemeral datastore is never locked.
3. The ephemeral portion of the intended configuration, applied state, and derived state does not persist over a reboot,
4. an ephemeral node cannot roll-back to its previous value,
5. Since ephemeral data store is just data that does not persist over a reboot, then in theory any node or group of nodes in a YANG data model could be ephemeral. The YANG data module must indicate what portion of the data model (if any) is ephemeral.
 - * A YANG data module could be all ephemeral (e.g. [I-D.ietf-i2rs-rib-data-model]) with no directly associated configuration models,
 - * A YANG model could be all ephemeral but associated with a configuration model
 - * or a single data node or data tree could be made ephemeral.
6. The management protocol (NETCONF/RESTCONF) needs to signal which portions of a data model(node, tree, or data model) are ephemeral in the module library [RFC7895].

4.3. I2RS Agent Caching of Ephemeral Data

The multiple control-plane datastore model [I-D.nmdsdt-netmod-revised-datastores] architecture allows multiple datastores which could allow an implementation of caching of ephemeral data in the I2RS Agent by having a main and a backup I2RS agent. Early implementations should at least support the single ephemeral data model, but MAY support the multiple datastore mode. It is important that these early implementations provide feedback for standardization on the following:

the policy knobs needed to make single ephemeral control planes datastores function,

the policy knobs needed to make multiple ephemeral control plane datastores which support caching work.

4.4. Massive Amounts of Configuration Data

Large amounts of data can flow from the I2RS agent to the I2RS client, or from the I2RS client to the I2RS Agent. The I2RS client may set or query ephemeral configuration in the routing system via the I2RS agent and receive operational state, notifications, or logging from the I2RS Agent on behalf of the I2RS routing system. I2RS Clients can send large amount of ephemeral configuration data to the I2RS Agent. The writes may be done via NETCONF (<edit-config> or an rpc function), or via RESTCONF (PUT, PATCH, POST). Reads can be done via NETCONF <get-config> or RESTCONF GET or query.

The I2RS RIB Data Model [I-D.ietf-i2rs-rib-data-model] also supports the use of rpc to add/delete RIBs, add/delete/update routes, and add/delete nexthops. If the I2RS client does a small to medium number of writes to the I2RS ephemeral state in the I2RS Agent in a routing system, the full validation that NETCONF or RESTCONF does will be able to be done without any reduction in speed to the I2RS high-performance system. For example, if the I2RS RIB Data Model has adds a 1000 routes, the I2RS RIB use of rpc to add/delete/update routes should be able to provide a high-performance system. Alternatively the NETCONF <edit-config> could update these 1000 routes with a write, or the RESTCONF POST, PUT or PATCH should be able to add the 1000 routes.

If a large number of ephemeral routes or filters are written (updates or new) by the I2RS Client to the ephemeral state in the I2RS agent, one of the key issues for a high performance interface is the time it takes to validate routes. Due to this concern, the I2RS architecture was design to allow less than the full NETCONF or RESTCONF validation. The concept is that the I2RS routes would be validated within the I2RS client and sent via a 99.999% reliable connection. In this scenario, the I2RS Agent would trust the validation that the I2RS Client did, and the communication of the route additions via the network connection.

An experiment regarding this has been done with the ODL code base update of ephemeral routes, but additional experimentation needs to be done prior to finalizing this design. Section 3.4.2 reviews how this process might be done, but many open issues exist in implementing this "low-validation" interface. Without additional experimentation and prototype code, this type of "low-validation",

4.5. Write Error handling

This section reviews I2RS normal error handling and error handling for rpc with no validation checks.

4.5.1. Normal validation checks

An I2RS agent validates an I2RS client's information by examining the following:

- o message syntax validation,
- o syntax validation for nodes of data model,
- o referential checks (leafref checks MUST clauses, and instance identifier),
- o checks groups of data within a data model or groups of data across data models,
- o write access to data,
- o if write access and values already exist, if I2RS client write access is higher than existing priority.

4.5.1.1. Reduced Validation (Experimental)

Can the I2RS protocol allow for reduced error checking? The need for speed in the I2RS protocol insertions in to the I2RS RIB suggest that it is worth experimenting for reduced validation in order to obtain high levels of throughput. If NETCONF or RESTCONF streams pre-checked routes to the datastore, what happens? Implementation experience is needed to determine the feasibility of this approach.

This feature may require a operator-applied policy knob swith a "no validation" feature

- o operator-applied policy knob enabling this feature;
- o rpc in a data model with the yang "ephemeral-validation no-check;"

4.6. IPFIX for traffic monitoring

Due to the potentially large data flow the traffic measurment statistics generate, these statistics are best handled by publication techniques within NETCONF or a separate protocol such as IPFIX. In the future version of the I2RS protocol may desire to support a data

stream outbound from the I2RS Agent to an I2RS client via the IPFIX protocol.

4.7. Binary encoding of RESTCONF/NETCONF

The binary encoding of JSON or XML encodnig in RESTCONF or NETCONF may provide a better throughput. Research needs to be done on what is the appropriate binary encoding.

4.8. Ephemeral state in DDoS environments

I2RS ephemeral state may operate in places where there is a DDoS attacks where the network devices are attacked. Is one attack plane the ability to remove all tracing if the I2RS reboots an attack vector?

5. Yang Changes

The data modules supporting the ephemeral datastore can use the Yang module library to describe their datastore.

The following key word must be able to specify ephemeral

```
ephemeral true;
```

Nice to have features:

It would be helpful for implementation of I2RS ephemeral data models to determine if the I2RS protocol feature set can support the I2RS data model needs. For this reason, it is helpful to group protocol features into "versions" and to put flags in the data model. At this point, the best place to put the summary of features is in an data model which defines these features. The discussion between implementers should be whether it is useful to have this features in some general yang location. An example of features that might be needed are:

- o i2rs version indicator;
- o i2rs transport-nonsecure "ok-to-use";
- o i2rs ephemeral-validation nocheck;
- o I2rs caching

6. IANA Considerations

This is a protocol strawman - nothing is going to IANA.

7. Security Considerations

The security requirements for the I2RS protocol are covered in [I-D.ietf-i2rs-protocol-security-requirements]. The security environment the I2RS protocol is covered in [I-D.ietf-i2rs-security-environment-reqs]. Any person implementing or deploying the I2RS protocol should consider both security requirements.

8. Acknowledgements

This document good work arises out of discussions with many experts in NETCONF, NETMOD, and I2RS WGs including:

- o Alia Atlas,
- o Ignas Bagdonas,
- o Andy Bierman,
- o Alex Clemm,
- o Eric Voit,
- o Kent Watsen,
- o Jeff Haas,
- o Russ White,
- o Keyur Patel,
- o Hariharan Ananthakrishnan,
- o Dean Bogdanavich,
- o Anu Nair,
- o Juergen Schoenwaelder, and
- o Kent Watsen.

Any errors or assumptions should be blamed on the authors, and not these experts.

9. References

9.1. Normative References:

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, DOI 10.17487/RFC4107, June 2005, <<http://www.rfc-editor.org/info/rfc4107>>.
- [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <<http://www.rfc-editor.org/info/rfc4960>>.
- [RFC5339] Le Roux, J.L., Ed. and D. Papadimitriou, Ed., "Evaluation of Existing GMPLS Protocols against Multi-Layer and Multi-Region Networks (MLN/MRN)", RFC 5339, DOI 10.17487/RFC5339, September 2008, <<http://www.rfc-editor.org/info/rfc5339>>.
- [RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <<http://www.rfc-editor.org/info/rfc5424>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<http://www.rfc-editor.org/info/rfc6020>>.
- [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>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<http://www.rfc-editor.org/info/rfc6242>>.
- [RFC6244] Shafer, P., "An Architecture for Network Management Using NETCONF and YANG", RFC 6244, DOI 10.17487/RFC6244, June 2011, <<http://www.rfc-editor.org/info/rfc6244>>.

- [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>>.
- [RFC7158] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7158, DOI 10.17487/RFC7158, March 2014, <<http://www.rfc-editor.org/info/rfc7158>>.
- [RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <<http://www.rfc-editor.org/info/rfc7589>>.
- [RFC7803] Leiba, B., "Changing the Registration Policy for the NETCONF Capability URNs Registry", BCP 203, RFC 7803, DOI 10.17487/RFC7803, February 2016, <<http://www.rfc-editor.org/info/rfc7803>>.
- [RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <<http://www.rfc-editor.org/info/rfc7895>>.
- [RFC7920] Atlas, A., Ed., Nadeau, T., Ed., and D. Ward, "Problem Statement for the Interface to the Routing System", RFC 7920, DOI 10.17487/RFC7920, June 2016, <<http://www.rfc-editor.org/info/rfc7920>>.
- [RFC7921] Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", RFC 7921, DOI 10.17487/RFC7921, June 2016, <<http://www.rfc-editor.org/info/rfc7921>>.
- [RFC7922] Clarke, J., Salgueiro, G., and C. Pignataro, "Interface to the Routing System (I2RS) Traceability: Framework and Information Model", RFC 7922, DOI 10.17487/RFC7922, June 2016, <<http://www.rfc-editor.org/info/rfc7922>>.
- [RFC7923] Voit, E., Clemm, A., and A. Gonzalez Prieto, "Requirements for Subscription to YANG Datastores", RFC 7923, DOI 10.17487/RFC7923, June 2016, <<http://www.rfc-editor.org/info/rfc7923>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<http://www.rfc-editor.org/info/rfc7950>>.

- [RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <<http://www.rfc-editor.org/info/rfc7952>>.
- [RFC7958] Abley, J., Schlyter, J., Bailey, G., and P. Hoffman, "DNSSEC Trust Anchor Publication for the Root Zone", RFC 7958, DOI 10.17487/RFC7958, August 2016, <<http://www.rfc-editor.org/info/rfc7958>>.

9.2. Informative References

- [I-D.ietf-i2rs-ephemeral-state]
Haas, J. and S. Hares, "I2RS Ephemeral State Requirements", draft-ietf-i2rs-ephemeral-state-22 (work in progress), November 2016.
- [I-D.ietf-i2rs-protocol-security-requirements]
Hares, S., Migault, D., and J. Halpern, "I2RS Security Related Requirements", draft-ietf-i2rs-protocol-security-requirements-17 (work in progress), September 2016.
- [I-D.ietf-i2rs-rib-data-model]
Wang, L., Ananthakrishnan, H., Chen, M., amit.dass@ericsson.com, a., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-06 (work in progress), July 2016.
- [I-D.ietf-i2rs-rib-info-model]
Bahadur, N., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-09 (work in progress), July 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-i2rs-yang-l3-topology]
Clemm, A., Medved, J., Varga, R., Tkacik, T., Liu, X., Bryskin, I., Guo, A., Ananthakrishnan, H., Bahadur, N., and V. Beeram, "A YANG Data Model for Layer 3 Topologies", draft-ietf-i2rs-yang-l3-topology-04 (work in progress), September 2016.

- [I-D.ietf-netconf-call-home]
Watsen, K., "NETCONF Call Home and RESTCONF Call Home",
draft-ietf-netconf-call-home-17 (work in progress),
December 2015.
- [I-D.ietf-netconf-keystore]
Watsen, K. and G. Wu, "Keystore Model", draft-ietf-
netconf-keystore-00 (work in progress), October 2016.
- [I-D.ietf-netconf-netconf-event-notifications]
Prieto, A., Clemm, A., Voit, E., Nilsen-Nygaard, E.,
Tripathy, A., Chisholm, S., and H. Trevino, "NETCONF
Support for Event Notifications", draft-ietf-netconf-
netconf-event-notifications-01 (work in progress), October
2016.
- [I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
Protocol", draft-ietf-netconf-restconf-18 (work in
progress), October 2016.
- [I-D.ietf-netconf-restconf-client-server]
Watsen, K. and J. Schoenwaelder, "RESTCONF Client and
Server Models", draft-ietf-netconf-restconf-client-
server-01 (work in progress), November 2016.
- [I-D.ietf-netconf-rfc5277bis]
Clemm, A., Prieto, A., Voit, E., Nilsen-Nygaard, E.,
Tripathy, A., Chisholm, S., and H. Trevino, "Subscribing
to Event Notifications", draft-ietf-netconf-rfc5277bis-01
(work in progress), October 2016.
- [I-D.ietf-netconf-yang-patch]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Patch
Media Type", draft-ietf-netconf-yang-patch-13 (work in
progress), November 2016.
- [I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-
Nygaard, E., Bierman, A., and B. Lengyel, "Subscribing to
YANG datastore push updates", draft-ietf-netconf-yang-
push-04 (work in progress), October 2016.
- [I-D.ietf-netconf-zerotouch]
Watsen, K. and M. Abrahamsson, "Zero Touch Provisioning
for NETCONF or RESTCONF based Management", draft-ietf-
netconf-zerotouch-11 (work in progress), October 2016.

[I-D.ietf-netmod-schema-mount]

Bjorklund, M. and L. Lhotka, "YANG Schema Mount", draft-ietf-netmod-schema-mount-03 (work in progress), October 2016.

[I-D.ietf-netmod-syslog-model]

Wildes, C. and K. Koushik, "A YANG Data Model for Syslog Configuration", draft-ietf-netmod-syslog-model-11 (work in progress), November 2016.

[I-D.nmdsdt-netmod-revised-datastores]

Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "A Revised Conceptual Model for YANG Datastores", draft-nmdsdt-netmod-revised-datastores-00 (work in progress), October 2016.

Authors' Addresses

Susan Hares
Huawei
Saline
US

Email: shares@ndzh.com

Amit Daas
Ericsson

Email: amit.dass@ericsson.com

I2RS working group
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2017

S. Hares
Huawei
S. Kini
Ericsson
L. Dunbar
Huawei
R. Krishnan
Dell
D. Bogdanovic
Juniper Networks
R. White
Linkedin
March 13, 2017

Filter-Based RIB Data Model
draft-ietf-i2rs-fb-rib-data-model-01

Abstract

This document defines a data model to support the Filter-based Routing Information Base (RIB) Yang data models. A routing system uses the Filter-based RIB to program FIB entries that process incoming packets by matching on multiple fields within the packet and then performing a specified action on it. The FB-RIB can also specify an action to forward the packet according to the FIB entries programmed using the RIBs of its routing instance.

The Filter based RIB is a protocol independent data structure which can be deployed in a configuration datastore, an ephemeral control plane data store.

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 14, 2017.

Copyright Notice

Copyright (c) 2017 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
1.1. Definition of Filter Based RIB	2
2. Requirements Language	4
3. Definitions and Acronyms	4
4. High level Yang structure for the FB-RIB	5
4.1. Top Level Yang Structure for ietf-fb-rib	7
4.2. Filter-Based RIB structures	8
5. yang models	9
5.1. Filter-Based RIB types	9
5.2. FB-RIB	16
6. IANA Considerations	18
7. Security Considerations	19
8. References	19
8.1. Normative References:	19
8.2. Informative References	19
Authors' Addresses	20

1. Introduction

This document provides a protocol-independent yang module for Filter Based Routing (FB-RIB) routing filters within a routing element. The informational model for this FB-RIB is in [I-D.ietf-i2rs-fb-rib-info-model].

1.1. Definition of Filter Based RIB

Filter-based routing is a technique used to make packet forwarding decisions based on a filter that is matched to the incoming packets and the specified action. It should be noted that that this is distinct from the static routes in the RIB where the routing is destination address based.

A Filter-Based RIB (Routing Information Base) is contained in a routing instance. It contains a list of filters (match-action conditions) and a list of interfaces the filter-based forwarding operates on, and default RIB(s).

A Filter Based RIB uses packet forwarding policy. If packet reception is considered an event, then the Filter-based RIB uses a minimalistic Event-matchCondition-Action policy with the following characteristics:

event = packet/frame received,

match condition - match on field in frame/packet or circumstances relating to packet reception (e.g. time received),

action - modify packet and forward/drop packet.

A Filter-based RIB entry specifies match filters for the fields in a packet (which may include layer 1 to layer 3 header fields, transport or application fields) or size of the packet or interface received on. The matches are contained in an ordered list of filters which contain pairs of match condition-action (aka event-condition-action).

If all matches fail, default action is to forward the packet using Destination Based forward from the default RIB(s). The default RIBs can be:

- o created by the I2RS Routing Information Base (RIB) manager using the yang model described in: in [I-D.ietf-i2rs-rib-info-model], or
- o configured RIB created using static routes or [I-D.ietf-netmod-routing-cfg].

Actions in the condition-action pair may impact forwarding or set something in the packet that will impact forwarding. Policy actions are typically applied before applying QoS constraints since policy actions may override QoS constraint.

The Filter-Based RIB can reside in the configuration datastore, a control plane datastore, or an ephemeral control plane data store (e.g. I2RS ephemeral control plane datastore).

The Interface to the Routing System (I2RS) [RFC7921] architecture provides dynamic read and write access to the information and state within the routing elements. The I2RS client interacts with the I2RS agent in one or more network routing systems. The I2RS architecture defines the I2RS control plane datastore as ephemeral - which means it does not persist across a reboot.

2. Requirements Language

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

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

3. Definitions and Acronyms

CLI

Command Line Interface

FB-RIB

Filter-Based Routing Information Base

FB-Route

The policy rules in the filter-based RIB are prescriptive of the Event-Condition-Action form which is often represented by if Condition then action".

Policy Group

Policy Groups are groups of policy rules. The groups of policy in the basic network policy [I-D.ietf-i2rs-pkt-eca-data-model] allow grouping of policy by name. This structure allow easier management of customer-based or provider based filters, but does not change the policy-rules list.

RIB IM

RIB Informational Model (RIB IM) [I-D.ietf-i2rs-rib-info-model]

Routing instance

A routing instance, in the context of the FB-FIB is a collection of RIBs, interfaces, and routing parameters. A routing instance creates a logical slice of the router and allows different logical slices; across a set of routers; to communicate with each other.

4. High level Yang structure for the FB-RIB

There are three levels in the Filter-Based RIB (FB-RIB) structure:

- o a global FB-RIB structures,
- o the common structure of the FB-RIB, and
- o the groupings that make up the FB-RIB

All structures have two types: configuration/ephemeral state and operational state.

This yang model allows for three types of FB-RIB installations in three types of datastores:

configuration (Config=TRUE, ephemeral=false, opstate definitions)

ephemeral control plane (E.g. I2RS Agent, config=TRUE, ephemeral=TRUE, opstate definitions), and

non-ephemeral control plane datastore (e.g. dBGp FB-FIB with config=TRUE; ephemeral=false, opstate which stores BGP Flow Specification received by bgp speaker from BGP peers).

Each of these cases is differentiated by using an "if-feature" to provide unique RIB under the routing instance.

Configuration RIBS

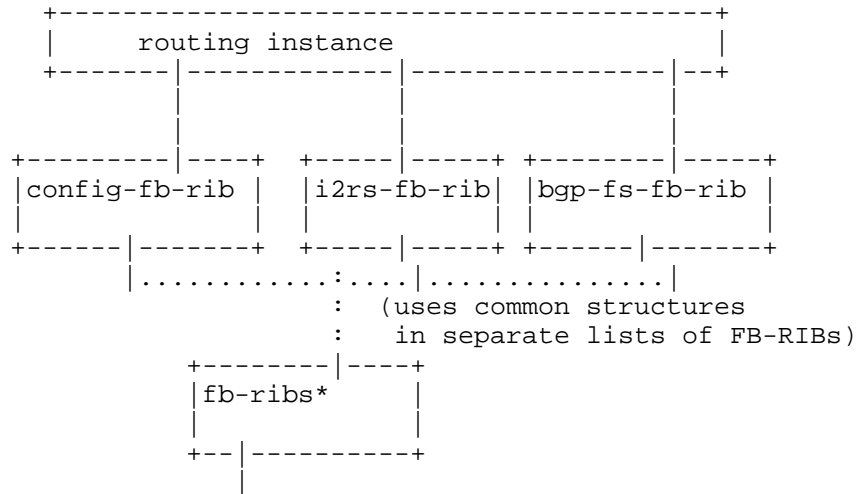


Figure 3: Routing instance with three types of Filter-FIB lists

The following section provides the high level yang structure diagrams for the following levels of structures for both config/ephemeral state and operationa.

- o ietf-fb-rib - contains filter-based RIBS for config, I2RS FB-RIB, and BGP Flow Specification.
- o fb-rib - that contains the structures for the filter-based grouping
- o fb-rib-types - that contains the structures for groupings within the filter-based RIBS

These structures are contained within the yang section in this draft.

The packet-reception ECA policy yang module is contained in the draft [I-D.ietf-i2rs-pkt-eca-data-model].

For those who desire more information regarding the logic behind the I2RS Filter-Based RIB, please see the Informational Model at: [I-D.ietf-i2rs-fb-rib-info-model].

4.1. Top Level Yang Structure for ietf-fb-rib

The Top-level Yang structure for a global FB-RIB types (similar to `acl`) is not defined for filter-based RIBs. The I2RS Filter-Based RIB should be defined under this structure under a routing instance. The three things under this RIB would be: configured Filter-Based RIB (aka Policy routing), I2RS reboot Ephemeral Filter-Based RIB, and BGP Flow Specification's Filter-Based RIB. All of these RIBs have similar actions.

There are two types top-level structures for ietf-fb-ribs: config and operational state.

The Top-level Yang structure for a global configuration of Filter-Based RIBs are:

```
Augments rt:logical-network-elements:\
    :logical-network-element:network-instances: \
        network-instance

ietf-fb-rib module
  +--rw ietf-fb-rib
    +--rw default-instance-name string
    +--rw default-router-id rt:router-id
    +--rw config-fb-ribs
      if-feature "config-filter-based-RIB";
      uses fb-ribs;
    +--rw i2rs-fb-ribs
      if-feature "I2RS-filter-based-RIB";
      uses fb-rib-t:fb-ribs;
    +--rw bgp-fs-fb-ribs
      if-feature "BGP-FS-filter-based-RIB";
      uses fb-rib-t:fb-ribs;
```

Figure 5: configuration state

The Top-level Yang structure for a global operational state of Filter-Based RIBs are:

```
Augments rt:logical-network-elements:\
      :logical-network-element:network-instances: \
        network-instance

ietf-fb-rib module
  +--rw ietf-fb-rib-opstate
    +--rw default-instance-name string
    +--rw default-router-id rt:router-id
    +--rw config-fb-rib-opstate
      if-feature "config-filter-based-RIB";
      uses fb-rib-t:fb-ribs-oper-status;
    +--rw i2rs-fb-rib-opstate {
      if-feature "I2RS-filter-based-RIB";
      uses fb-rib-t:fb-ribs-oper-status;
    +--rw bgp-fs-fb-rib-opstate
      if-feature "BGP-FS-filter-based-RIB";
      uses fb-rib-t:fb-ribs-oper-status;
```

Figure 5: operational state

4.2. Filter-Based RIB structures

The Top-level yang structures at the Filter-Based RIB level have two types: configuration and operational state.

The Top-level Yang structure for the FB-RIB types is:

```

module: fb-rib-types:
+--rw fb-ribs
  +--rw fb-rib* [rib-name]
    |   +--rw rib-name string
    |   |   rw fb-type identityref / ephemeral or not
    |   +--rw rib-afi rt:address-family
    |   +--rw fb-rib-intf* [name]
    |   |   +--rw name string
    |   |   +--rw intf if:interface
    |   +--rw default-rib
    |   |   +--rw rt-rib string
    |   |   +--rw config-rib string; // config rib name
    |   |   +--rw i2rs-rib:routing-instance:name
    |   |   +--rw i2rs-rib string; //ephemeral rib name
    |   |   +--rw bgp-instance-name string
    |   |   +--rw bgp-rib string //session ephemeral
    |   +--rw fb-rib-refs
    |   |   +--rw fb-rib-update-ref uint32
    |   |   |   /count of writes
    |   +--rw instance-using*
    |   |   device:networking-instance:\
    |   |   |   /networking-instance-name
    |   +--uses pkt-eca:pkt-eca-policy-set
    |   +--uses acls:access-lists

```

Figure 6: FB RIB Type Structure

Note: acls:access-lists is the list of ACL filters in [I-D.ietf-netmod-acl-model].

High Level Yang

```

+--rw fb-ribs-oper-status
  +--rw fb-rib-oper-status* [fb-rib-name]
    uses pkt-eca:pkt-eca-opstate

```

5. yang models

5.1. Filter-Based RIB types

```

<CODE BEGINS> file "ietf-fb-rib-types@2017-03-13.yang"
module ietf-fb-rib-types {

  yang-version "1";

  // namespace
  namespace "urn:ietf:params:xml:ns:yang:ietf-fb-rib-types";

```

```
prefix "fb-rib-t";
  import ietf-interfaces {prefix "if";}
  import ietf-routing {prefix "rt";}
  import ietf-pkt-eca-policy {prefix "pkt-eca";}
  import ietf-access-control-lists {prefix "acls";}

// meta
organization
  "IETF";

contact
  "email: shares@ndzh.com;
    email: sriganesh.kini@ericsson.com
    email: cengiz@packetdesign.com
    email: ivandean@gmail.org
    email: linda.dunbar@huawei.com;
    email: russ@riw.com;
  ";

description
  "This module describes a YANG model for the I2RS
  Filter-based RIB Types. These types
  specify types for the Filter-Based RIB.

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

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD
  License set forth in Section 4.c of the IETF Trust's Legal
  Provisions Relating to IETF Documents
  (http://trustee.ietf.org/license-info).";

revision "2017-03-13" {
  description
    "Filter-Based RIB protocol ";
  reference "draft-ietf-i2rs-fb-rib-data-model-01";
}

typedef fb-rib-type-def {
  type identityref {
    base "fb-rib-type";
  }
  description
    "This type is used to refer to
    source of Filter-Based RIB:"
```

```
        configuration, I2RS, Flow-Spec.";
    }

    identity fb-rib-type {
        description
            "This type is used to refer to
            source of Filter-Based RIB:
            configuration, I2RS, Flow-Spec.";
    }

    identity fb-rib-config-type {
        base fb-rib-type;
        description
            "config Filter-Based RIB";
    }

    identity fb-rib-i2rs-ephemeral-type {
        base fb-rib-type;
        description
            "I2RS Reboot ephemeral Filter-Based RIB";
    }

    identity fb-rib-BGP-FS-type {
        base fb-rib-type;
        description
            "BGP Flow Specification Filter-Based RIB";
    }

    typedef fb-rib-policy-type-def {
        type identityref {
            base "fb-rib-policy-type";
        }
        description
            "This type is used to refer to FB-RIB type";
    }

    identity fb-rib-policy-type {
        description
            "Types of filter-based policies
            acl and eca";
    }

    identity fb-rib-acl {
        base fb-rib-policy-type;
        description
            "filter based policy based on access-lists";
    }
```

```
    identity fb-bnp-eca-rules {
        base fb-rib-policy-type;
        description
        "filter based policy based on qos forwarding rules";
    }

typedef fb-rules-status {
    type identityref {
        base "fb-rule-opstat";
    }
    description
    "This type is used to refer to FB-RIB type";
}

identity fb-rule-opstat {
    description
    "operational statuses for filter rules
    inactive and active";
}

identity fb-rule-inactive {
    base fb-rule-opstat;
    description
    "policy rule is inactive";
}

identity fb-rule-active {
    base fb-rule-opstat;
    description
    "policy rule is active";
}

grouping fb-rib-rule-order-status {
    leaf statement-order {
        type uint16;
        description "order identifier";
    }
    leaf statement-oper_status {
        type fb-rules-status;
        description "status of rule";
    }
    description "filter-rib
    policy rule order and status";
}

grouping fb-rib-group-order-status {
    leaf group-refcnt {
        type uint16;
    }
}
```

```
        description "refcnt for this group";
    }
    leaf group-installed {
        type uint32;
        description "number of rules installed";
    }
    leaf group-matches {
        type uint64;
        description "number of matches by all
            rules in group";
    }
    description "fb-rib group list order
        and status info.";
}

grouping fb-rib-updates {
    leaf fb-rib-update-ref {
        type uint64;
        description
            "number of updates to this FB RIB
            since last reboot";
    }
    description "FB-RIB update info";
}

grouping default-fb-rib {
    // configuration instance for default RIB
    leaf config-instance {
        type string;
        description "instance name - string until
            netmod fixes mount issues";
    }
    leaf config-rib {
        type string;
        description "name of config default RIB";
    }
    //I2RS default instance for default RIB
    leaf i2rs-instance-name {
        type string;
        description "I2RS instance name";
    }
    leaf i2rs-rib-name {
        type string;
        description "name of default I2RS RIB";
    }
    leaf bgp-instance-name {
        type string;
        description "name of bgp instance";
    }
}
```



```
    }

    leaf bgp-fs-rib-name {
        type string;
        description "name of BGP
                    flow specification default RIB";
    }
    description "default RIB for forwarding
                if the policy match";
}

grouping fb-ribs {
    list fb-rib {
        key fb-rib-name;
        leaf fb-rib-name {
            type string;
            mandatory true;
            description "RIB name";
        }
        uses rt:address-family;
        leaf fb-type {
            type fb-rib-type-def;
            description "type of RIB
                        list: config, I2RS reboot
                        ephemeral, BGP Flow Specification
                        ephemeral. ";
        }
    }
    list fb-rib-intf {
        key "name";
        leaf name {
            type if:interface-ref;
            description
                "A reference to the name of a
                 configured network layer
                 interface.";
        }
        description "This represents
                    the list of interfaces
                    associated with this routing instance.
                    The interface list helps constrain the
                    boundaries of packet forwarding.
                    Packets coming on these interfaces are
                    directly associated with the given routing
                    instance. The interface list contains a
                    list of identifiers, with each identifier
                    uniquely identifying an interface.";
    }
    uses default-fb-rib; // defaults ribs
}
```

```
        uses fb-rib-updates; // write refs to this RIB
list instance-using {
    key instance-name;
    leaf instance-name {
        type string;
        description
            " name of instance using this fb-rib
            rt:routing-instance";
    }
    description "instances using
    this fb-rib";
}
// ordered rule list + group list
uses pkt-eca:pkt-eca-policy-set;

// ordered acl list
uses acs:access-lists;

description "Configuration of
an filter-based rib list";
}
description "fb-rib group";
}

grouping fb-ribs-oper-status {
    list fb-rib-oper-status {
        key fb-rib-name;
        leaf fb-rib-name {
            type string;
            description "rib name";
        }
        leaf pkt-eca-cfged {
            type boolean;
            description
                "pkt eca configured";
        }
        leaf acs-cfged {
            type boolean;
            description
                "acs configured";
        }
    }
    uses pkt-eca:pkt-eca-opstate;
    description
        "Configuration of
        an filter-based rib list";
}
description
    "list of FB-FIB operational
```

```
        status";
    }

}
```

<CODE ENDS>

5.2. FB-RIB

```
<CODE BEGINS> file "ietf-fb-rib@2017-03-13.yang"
module ietf-fb-rib {
    yang-version "1";

    // namespace
    namespace "urn:ietf:params:xml:ns:yang:ietf-fb-rib";
    // replace with iana namespace when assigned
    prefix "fb-rib";

    // import some basic inet types
    import ietf-yang-types {prefix "yang";}
    import ietf-fb-rib-types { prefix "fb-rib-t";}

    // meta
    organization
        "IETF";

    contact
        "email: sriganesh.kini@ericsson.com
          email: cengiz@packetdesign.com
          email: anoop@ieee.duke.edu
          email: ivandean@gmail.org
          email: shares@ndzh.com;
          email: linda.dunbar@huawei.com;
          email: russ@riw.com;
          ";

    description
        "This Top level module describes a YANG model for the I2RS
        Filter-based RIB which is an global protocol independent FB RIB module."
;

    revision "2017-03-13" {
        description "initial revision";
        reference "draft-ietf-i2rs-fb-rib-data-model-01";
    }

    feature config-filter-based-RIB {
```

```
description
  "This feature means that a node support
  config filter-based rib.";
}
  feature I2RS-filter-based-RIB {
description
  "This feature means that a node support
  I2RS filter-based rib.";
}
  feature BGP-FS-filter-based-RIB {
description
  "This feature means that a node support
  BGP FS filter-based rib.";
}

  container ietf-fb-rib {
    presence "top-level structure for
    configuration";
    leaf default-instance-name {
      type string;
      mandatory true;
    description
      "A routing instance is identified by its name,
      INSTANCE_name. This MUST be unique across all routing
      instances in a given network device.";
    }
    leaf default-router-id {
      type yang:dotted-quad;
      description "Default router id";
    }
    container config-fb-rib {
      if-feature config-filter-based-RIB;
      uses fb-rib-t:fb-ribs;
      description "config filter-based RIB";
    }

    container i2rs-fb-rib {
      if-feature I2RS-filter-based-RIB;
      uses fb-rib-t:fb-ribs;
      description "i2rs filter-based RIB";
    }
    container bgp-fs-fb-rib {
      if-feature BGP-FS-filter-based-RIB;
      uses fb-rib-t:fb-ribs;
      description "bgp fs filter-based RIB";
    }
    description "fb-rib augments routing instance";
  }
```

```
    }

    container ietf-fb-rib-opstate {
        presence "top-level structure for
        op-state";
        config "false";
    leaf default-instance-name {
        type string;
        mandatory true;
    description
        "A routing instance is identified by its name,
        INSTANCE_name. This MUST be unique across all routing
        instances in a given network device.";
    }
        leaf default-router-id {
            type yang:dotted-quad;
            description "Default router id";
        }
        container config-fb-rib-opstate {
            if-feature config-filter-based-RIB;
            uses fb-rib-t:fb-ribs-oper-status;
            description "config filter-based RIB";
        }
        container i2rs-fb-rib-opstate {
            if-feature I2RS-filter-based-RIB;
            uses fb-rib-t:fb-ribs-oper-status;
            description "bgp-fs filter-based RIB";
        }
        container bgp-fs-fb-rib-opstate {
            if-feature BGP-FS-filter-based-RIB;
            uses fb-rib-t:fb-ribs-oper-status;
            description "bgp fs filter-based RIB";
        }
        description "fb-rib augments routing instance";
    }
}
```

<CODE ENDS>

6. IANA Considerations

TBD

7. Security Considerations

A I2RS RIB is ephemeral data store that will dynamically change traffic paths set by the routing configuration. An I2RS FB-RIB provides dynamic Event-Condition-Action policy that will further change the operation of forwarding by allow dynamic policy and ephemeral RIBs to alter the traffic paths set by routing configuration. Care must be taken in deployments to use the appropriate security and operational control to make use of the tools the I2RS RIB and I2RS FB-RIB provide.

8. References

8.1. Normative References:

- [I-D.ietf-i2rs-pkt-eca-data-model]
Hares, S., Wu, Q., and R. White, "Filter-Based Packet Forwarding ECA Policy", draft-ietf-i2rs-pkt-eca-data-model-02 (work in progress), October 2016.
- [I-D.ietf-i2rs-rib-data-model]
Wang, L., Ananthakrishnan, H., Chen, M., amit.dass@ericsson.com, a., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-07 (work in progress), January 2017.
- [I-D.ietf-netmod-acl-model]
Bogdanovic, D., Koushik, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-10 (work in progress), March 2017.
- [I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-25 (work in progress), November 2016.

8.2. Informative References

- [I-D.ietf-i2rs-fb-rib-info-model]
Kini, S., Hares, S., Dunbar, L., Ghanwani, A., Krishnan, R., Bogdanovic, D., and R. White, "Filter-Based RIB Information Model", draft-ietf-i2rs-fb-rib-info-model-00 (work in progress), June 2016.

- [I-D.ietf-i2rs-rib-info-model]
Bahadur, N., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-10 (work in progress), December 2016.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC7921] Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", RFC 7921, DOI 10.17487/RFC7921, June 2016, <<http://www.rfc-editor.org/info/rfc7921>>.

Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Email: shares@ndzh.com

Sriganesh Kini
Ericsson

Email: sriganesh.kini@ericsson.com

Linda Dunbar
Huawei
USA

Email: linda.dunbar@huawei.com

Ram Krishnan
Dell

Email: Ramkri123@gmail.com

Dean Bogdanovic
Juniper Networks
Westford, MA

Email: ivandean@gmail.org

Russ White
Linkedin

Email: russ@riw.us

I2RS working group
Internet-Draft
Intended status: Standards Track
Expires: April 8, 2017

S. Hares
Q. Wu
Huawei
R. White
Ericsson
October 5, 2016

Filter-Based Packet Forwarding ECA Policy
draft-ietf-i2rs-pkt-eca-data-model-02.txt

Abstract

This document describes the yang data model for packet forwarding policy that filters received packets and forwards (or drops) the packets. Prior to forwarding the packets out other interfaces, some of the fields in the packets may be modified. If one considers the packet reception an event, this packet policy is a minimalistic Event-Match Condition-Action policy. This policy controls forwarding of packets received by a routing device on one or more interfaces on which this policy is enabled. The policy is composed of an ordered list of policy rules. Each policy rule contains a set of match conditions that filters for packets plus a set of actions to modify the packet and forward packets. The match conditions can match tuples in multiple layers (L1-L4, application), interface received on, and other conditions regarding the packet (size of packet, time of day). The modify packet actions allow for setting things within the packet plus decapsulation and encapsulation packet. The forwarding actions include forwarding via interfaces, tunnels, or nexthops and dropping the packet. The policy model can be used with the session ephemeral (BGP Flow Specifications), reboot ephemeral state (I2RS ephemeral), and non-ephemeral routing/forwarding state (e.g. configuration state).

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 8, 2017.

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
1.1. Definitions and Acronyms	3
1.2. Antecedents this Policy in IETF	4
2. Generic Route Filters/Policy Overview	4
3. BNP Rule Groups	5
4. Packet ECA (event-condition-action) Filter Model in High Level Yang	7
4.1. modules included	7
4.2. top level description	8
4.3. Conditional filters	10
4.3.1. Event Match Filters	11
4.3.2. ECA Packet Condition Matches	12
4.4. ECA Packet Actions	19
4.5. Policy Conflict Resolution strategies	22
4.6. External Data	22
5. i2rs-eca-policy Yang module	23
6. IANA Considerations	53
7. Security Considerations	54
8. Informative References	54
Authors' Addresses	55

1. Introduction

This document describes the yang data model for packet forwarding policy that filters received packets and forwards (or drops) the packets. Prior to forwarding the packets out other interfaces, some of the fields in the packets may be modified. If one considers the reception of a packet as an event, this minimalistic Event-Match Condition-Action policy. If one considers the reception of packets

containing Layer 1 to Layer 4 + application data a single packet, then this minimalistic policy can be called a packet-only ECA policy. This document will use the term packet-only ECA policy for this model utilizing the term "packet" in this fashion.

This packet-only ECA policy data model supports an ordered list of ECA policy rules where each policy rule has a name. The match condition filters include matches on

- o content of packet headers for layer 1 to layer 4,
- o application protocol data and headers,
- o interfaces the packet was received on,
- o time packet was received, and
- o size of packet.

The actions include packet modify actions and forwarding options. The modify options allow for the following:

- o setting fields in the packet header at Layer 2 (L2) to Layer 4 (L4), and
- o encapsulation and decapsulation the packet.

The forwardingng actions allow forwardsing the packet via interfaces, tunnels, next-hops, or dropping the packet. setting things within the packet at Layer 2 (L2) to layer 4 (L4) plus overlay or application data.

The first section of this draft contains an overview of the policy structure. The second provides a high-level yang module. The third contains the yang module.

The high-level yang and the actual yang are not aligned. This is an interim-release of this document.

1.1. Definitions and Acronyms

INSTANCE: Routing Code often has the ability to spin up multiple copies of itself into virtual machines. Each Routing code instance or each protocol instance is denoted as Foo_INSTANCE in the text below.

NETCONF: The Network Configuration Protocol

PCIM - Policy Core Information Model

RESTconf - http programmatic protocol to access yang modules

1.2. Antecedents this Policy in IETF

Antecedents to this generic policy are the generic policy work done in PCIM WG. The PCIM work contains a Policy Core Information Model (PCIM) [RFC3060], Policy Core Informational Model Extensions [RFC3460] and the Quality of Service (QoS) Policy Information Model (QPIM) ([RFC3644]) From PCIM comes the concept that policy rules which are combined into policy groups. PCIM also refined a concept of policy sets that allowed the nesting and aggregation of policy groups. This generic model did not utilize the concept of sets of groups, but could be expanded to include sets of groups in the future.

2. Generic Route Filters/Policy Overview

This generic policy model represents filter or routing policies as rules and groups of rules.

The basic concept are:

Policy set:

Policy set is a set of policies

Policy:

A policy is a is an ordered set of rules .

Rule

A Rule is represented by the semantics "If Condition then Action".
A Rule may have a priority assigned to it.

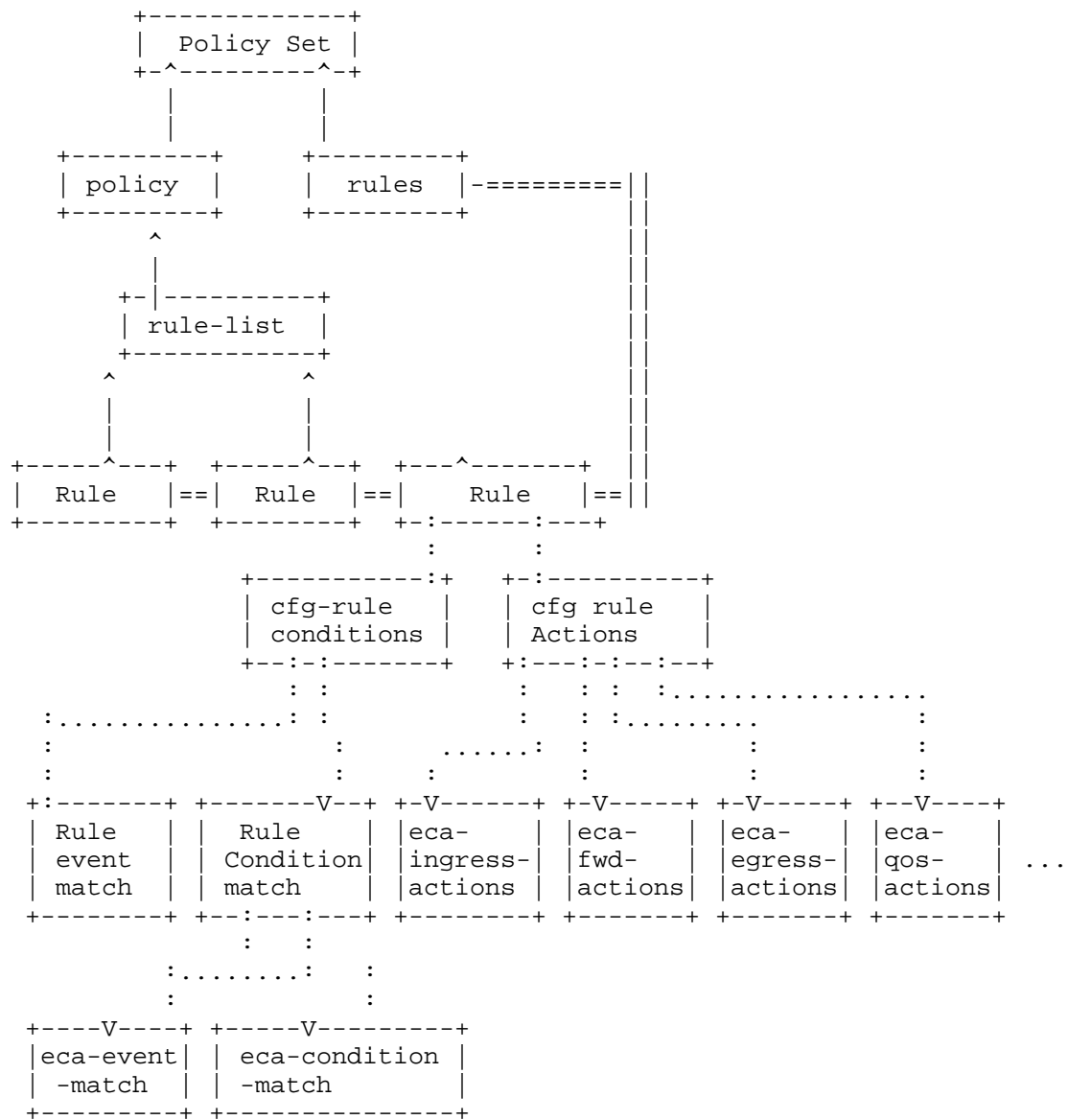


Figure 1: ECA rule structure

3. BNP Rule Groups

The pkt ECA policy is a policy which is an ordered set of pkt-ECA policy rules. The rules assume the event is the reception of a packet on the machine on a set of interfaces. This policy is

associated with a set of interfaces on a routing device (physical or virtual).

A Policy allows for the easy combination of rules for management stations or users. A policy has the following elements:

- o name that identifies the grouping of policy rules
- o module reference - reference to a yang module(s) in the yang module library that this group of policy writes policy to
- o list of rules

A policy group may have rules at different order levels. For example, policy group 1 could have three policy rules at rule order 1 and four policy rules at rule order 5.

The rule has the following elements: name, order, status, priority, reference cnt, and match condition, and action as shown as shown in figure 2. The order indicates the order of the rule within the the complete list. The status of the rule is (active, inactive). The priority is the priority within a specific order of policy/filter rules. A reference count (refcnt) indicates the number of entities (E.g. network modules) using this policy. The generic rule match-action conditions have match operator, a match variable and a match value. The rule actions have an action operator, action variable, and an action value.

Rules can exist with the same rule order and same priority. Rules with the same rule order and same priority are not guaranteed to be at any specific ordering. The order number and priority have sufficient depth that administrators who wish order can specify it.

The generic match conditions are specific to a particular layer are refined by matches to a specific layer (as figure 2 shows), and figure 5's high-level yang defines. The general actions may be generic actions that are specific to a particular layer (L1, L2, L3, service layer) or time of day or packet size. The qos actions can be setting fields in the packet at any layer (L1-L4, service) or encapsulating or decapsulating the packet at a layer. The fwd-actions are forwarding functions that forward on an interface or to a next-hop. The rule status is the operational status per rule.

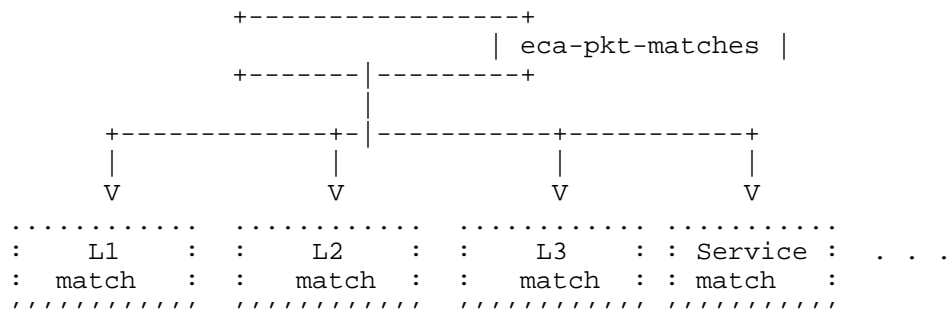


Figure 2 match logic

4. Packet ECA (event-condition-action) Filter Model in High Level Yang

The description of packet event-condition-action data model include:

module included in module

top level diagram

4.1. modules included

Below is the high level module inclusions.

```

module:pkt-eca-policy
  import ietf-inet-types {prefix "inet"}
  import ietf-interface {prefix "if"}
  import ietf-i2rs-rib {prefix "i2rs-rib"}

  import ietf-interfaces {
    prefix "if";
  }
  import ietf-inet-types {
    prefix inet;
    //rfc6991
  }

  import ietf-i2rs-rib {
    prefix "i2rs-rib";
  }
  
```

Figure 3 - high level inclusion

4.2. top level description

Below is the high level yang diagram

```

module ietf-pkt-eca-policy
  +--rw pkt-eca-policy-cfg
  |   +--rw pkt-eca-policy-set
  |   |   +--rw policies* [policy-name]
  |   |   |   +--rw policy-name string
  |   |   |   +--rw vrf-name string
  |   |   |   +--rw address-family
  |   |   |   +--rw rule-list* [rule-name]
  |   |   |   |   +--rw rule-name
  |   |   |   |   +--rw rule-order-id uint16
  |   |   |   |   +--rw default-action-id integer
  |   |   |   |   +--rw default-resolution-strategy-id integer
  |   |   +--rw rules* [order-id rule-name]
  |   |   |   +--rw order-id uint16
  |   |   |   +--rw rule-name string
  |   |   |   +--rw policy-name string
  |   |   |   +--rw cfg-rule-conditions [rule-cnd-id]
  |   |   |   |   +--rw rule-cnd-id uint32
  |   |   |   |   +--rw support
  |   |   |   |   |   +--rw event-matches boolean
  |   |   |   |   |   +--rw pkt-matches boolean
  |   |   |   |   |   +--rw usr-context-matches boolean
  |   |   |   |   +--rw eca-events-match* [rule-event-id]
  |   |   |   |   |   +--rw rule-event-it uint16
  |   |   |   |   |   |   ... time-event match (see below)
  |   |   |   |   +--rw eca-condition-match
  |   |   |   |   |   +--rw eca-pkt-matches* [pkt-match-id]
  |   |   |   |   |   |   ... (see packet matches below)
  |   |   |   |   |   |   ... (address, packet header, packet payload)
  |   |   |   |   |   +--rw eca-user-context-matches* [usr-match-id]
  |   |   |   |   |   |   ... (see user context match below)
  |   |   |   +--rw cfg-rule-actions [cfgr-action-id]
  |   |   |   |   +--rw cfgr-action-id
  |   |   |   |   +--rw eca-actions* [action-id]
  |   |   |   |   |   +--rw action-id uint32
  |   |   |   |   |   +--rw eca-ingress-actions*
  |   |   |   |   |   |   ... (permit, deny, mirror)
  |   |   |   |   |   +--rw eca-fwd-actions*
  |   |   |   |   |   |   ... (invoke, tunnel encap, fwd)
  |   |   |   |   |   +--rw eca-egress-acttions*
  |   |   |   |   |   |   ...
  |   |   |   |   +--rw eca-qos-actions*
  |   |   |   |   |   ...

```



```

|         | |   +--rw eca-security-actions*
+--rw policy-conflict-resolution* [strategy-id]
|   +--rw strategy-id integer
|   +--rw filter-strategy identityref
|   |   .. FMR, ADTP, Longest-match
+--rw global-strategy identityref
+--rw mandatory-strategy identityref
+--rw local-strategy identityref
+--rw resolution-fcn uint32
+--rw resolution-value uint32
+--rw resolution-info string
+--rw associated-ext-data*
|   |   +--rw ext-data-id integer
+--rw cfg-external-data* [cfg-ext-data-id]
|   +--rw cfg-ext-data-id integer
|   +--rw data-type integer
|   +--rw priority uint64
|   |   uses external-data-forms
|   ... (other external data)
+--rw pkt-eca-policy-opstate
+--rw pkt-eca-opstate
+--rw policies-opstat* [policy-name]
|   +--rw rules-installed;
|   +--rw rules_opstat* [rule-name]
|   |   +--rw strategy-used [strategy-id]
+--rw rules_opstate* [rule-order rule-name]
|   +--rw status
|   +--rw rule-inactive-reason
|   +--rw rule-install-reason
|   +--rw rule-installer
|   +--rw refcnt
+--rw rules_pktstats* [rule-order rule-name]
|   +--rw pkts-matched
|   +--rw pkts-modified
|   +--rw pkts-forward
|       +--rw op-external-data [op-ext-data-id]
|       |   +--rw op-ext-data-id integer
|       |   +--rw type identityref
|       |   +--rw installed-priority integer
|       |   |   (other details on external data )

```

figure 4 - high-level yang for policy set

The three levels of policy are expressed as:

```
Config Policy definitions
=====
Policy level: pkt-eca-policy-set
group level:  pkt-eca-policy-set:policies
rule level:   pkt-eca-policy-set:rules
external id:  pkt-eca-policy-set:cfg-external-data

Operational State for Policy
=====
Policy level: pkt-eca-policy-opstate
group level:  pkt-eca-opstate:policies-opstat*
rule level:   pkt-eca-opstate:rules_opstat*
               pkt-eca-opstate:rules-pktstat*
external id:  pkt-eca-opstate:op-external-data*
```

figure 5

4.3. Conditional filters

The condition filters in the packet eca policy module included the following:

- o event filters - time as an augment to reception of a packet.
- o conditional matches on packet content or user-related content

The sections below provide the high-level yang for these sections of the model.

```

module:i2rs-pkt-eca-policy
.....
+--rw pkt-eca-policy-cfg
|   +--rw pkt-eca-policy-set
|   +--rw pkt-eca-policy-set
|   |   ...
|   +--rw rules* [order-id rule-name]
|       +--rw order-id uint16
|       +--rw rule-name string
|       +--rw policy-name string
|       +--rw cfg-rule-conditions [rule-cnd-id]
|           +--rw rule-cnd-id integer
|           +--rw eca-events-match* [rule-event-id]
|               +--rw rule-event-it uint16
|               |   ... time-event match (see below)
|       +--rw eca-condition-match
|           +--rw eca-pkt-matches* [pkt-match-id]
|               ... (see L1-L4 matches below)
|           +--rw eca-usr-context-matches* [usr-match-id]
|               (user, schedule, region, target,
|               state, direction)

```

Figure 6

4.3.1. Event Match Filters

The default event is the event of receiving a packet. In addition, the events allow a time-event match. Time events are provided as a list which includes specific times or ranges of time.

```

|  +--rw pkt-eca-policy-set
|  +--rw pkt-eca-policy-set
|  | ...
|  +--rw rules* [order-id rule-name]
|  |   +--rw order-id uint16
|  |   +--rw rule-name string
|  |   +--rw policy-name string
|  |   +--rw cfg-rule-conditions [rule-cnd-id]
|  |   |   +--rw rule-cnd-id  uint32
|  |   |   +--rw support
|  |   |   |   +--rw event-matches boolean
|  |   |   |   +--rw pkt-matches boolean
|  |   |   |   +--rw usr-context-matches boolean
|  |   |   +--rw eca-events-match* [rule-event-id]
|  |   |   |   +--rw rule-event-it  uint16
|  |   |   |   +--rw time-type identityref
|  |   |   |   +--(one-time)
|  |   |   |   |   +--rw event-time yang:date-and-time
|  |   |   |   +--(range-time)
|  |   |   |   +--rw event-start-time yang:date-and-time
|  |   |   |   +--rw event-end-time yang:date-and-time

```

figure 7

4.3.2. ECA Packet Condition Matches

The ECA condition matches are packet matches (eca)

4.3.2.1. Packet-match filter list (eca-pkt-match*)

The packet match content filters include: address filters and packet header content filters, and packet payload filters.

```

module:i2rs-pkt-eca-policy
+--pkt-eca-policy-set
  +--rw rules* [order-id rule-name]
    | ....
    | +--rw cfg-rule-conditions [rule-cnd-id]
    |   +--rw rule-cnd-id uint32
    |   +--rw support
    |     +--rw event-matches boolean
    |     +--rw pkt-matches boolean
    |     +--rw usr-context-matches boolean
    |   +--rw eca-event-match
    |     ...
    | +--rw eca-condition-match
    | +--rw eca-pkt-matches* [pkt-match-id]
    |   +--rw packet-match-id uint16
    |   +--rw packet-match-type identityref
    |   +--(packet-match-type)?
    |     +--:(address-pkt-match)
    |       | ...
    |       +--:(layer-pkt-match)
    |         | ...
    |         +--:(payload-pkt-match)
    |           | ...
    | +--rw eca-user-matches [user-match-id]

```

Figure 8

4.3.2.1.1. Match filters for addresses in packet

The address matches match the L3, mpls, MAC and interface address scope. Figure x shows this match

```

+--rw eca-pkt-matches* [pkt-match-id]
|   +--rw packet-match-id uint16
|   +--rw eca-pkt-match-type identityref
|   +--address-scope?
|   |   +--:(route-type)
|   |   |   +--:(ipv4)
|   |   |   |   ... src, dest, src-dest
|   |   |   +--:(ipv6)
|   |   |   |   ... src, dest, src-dest
|   |   |   +--:(mpls)
|   |   |   |   ... 32 bit label
|   |   |   +--:(mac)
|   |   |   |   ... src, dest, src-dest
|   |   |   +--:(interface-route)
|   |   |   |   .... interface

```

Figure 9

4.3.2.1.2. Packet header matches

The packet header matches match interface, L1-L4 headers, service chain headers, and packet size. The L1 header expected to be a null match except if there is an advanced L1 technology such as l1 with a L1 identifier that can be detected in the packet. Figure x shows these matches.

```

+--rw (layer-type)
+--:(interface-match-type)
|   ...
+--:(L1-header-match)
|   ...
+--:(L2-header-match)
|   +--(802.1Q)
|   |   ...
|   +--(802.11)
|   |   ...
|   +--(802.15)
|   |   ...
|   +--(NVGRE)
|   |   ...
|   +--(VXLAN)
|   |   ...
|   +--(MPLS )
|   |   ..
+--:(L3-header-match)
|   +--(l3-ipv4-header)
|   |   ...
|   +--(l3-ipv6-header)
|   |   ...
|   +--(l3-gre-header)
|   |   ...
+--: L4-header-match
|   +--(l4-tcp-header)
|   |   ...
|   +--(l4-udp-header)
|   |   ...
|   +--(l4-sctp-header)
|   |   ...
+--: Service-header-match
|   +--(sf-chain-meta-match)
|   |   ...
|   +--(sf-path-meta-match)
|   |   ..
+--:(packet-size)
|   +--l1-size-match uint32
|   +--l2-size-match uint32
|   +--l3-size-match uint32
|   +--l4-size-mtach uint32
|   +--service-meta-size uint32
|   +--leaf service-meta-payload uint32
+---rw packet

```

Figure 10

4.3.2.1.3. Payload matches

The payload information is a stream of bytes to be found in the packet payload beyond the L4 or service-path header. The structure of this data is simply a list of byte strings as figure x shows.

```

| | | | |....
| | | | |---rw eca-pkt-matches* [pkt-match-id]
| | | | |   |--rw packet-match-id uint16
| | | | |   |--rw packet-match-type identityref
| | | | |   +--(packet-match-type)?
| | | | |     +---:(address-pkt-match)
| | | | |       | ...
| | | | |     +---:(layer-pkt-match)
| | | | |       | ...
| | | | |     +---:(payload-pkt-match)
| | | | |       |--rw packet-payload *[packet-payload-id]
| | | | |       |--rw packet-payload-id  uint16
| | | | |       |--rw payload-match-bytes uint16
| | | | |       |--rw packet-payload  string

```

Figure 11

4.3.2.2. Matches on User Context

The match on user context allows filtering for a packet plus a filter related to a user.

Since not all I2RS routers are access routers, the support for matches has a flag for user filter. It is expected that core routers may not support contextual matching.

One example of user filters is for is parental controls. During school hours, the teenager Joe is restrict from certain web sites from September 1 while Joe is at at school. This "school" filter is an example of a period filter which has a start time (8:00am) and end time (3:30pm), which is valid beginning September 1, 2016. This filter applies only to the region of school networks. The filter looks for specific entertainment (e.g. YouTube) web sites, social-media (e.g. facebook), and gaming sites. This block is for their mobile phone and tablet, but not the computer that says at home.

The following is the components

- o user identifier (name, tenant id, virtual network),
- o schedule for the user filter (once or periodic, time range (start/end), and weekly validity check),

- o region this filter is valid.
- o targeted services, applications, devices, and state.

```

module:i2rs-pkt-eca-policy
.....
+--rw pkt-eca-policy-cfg
|
+--rw pkt-eca-policy-set
+--rw pkt-eca-policy-set
| ...
+--rw rules* [order-id rule-name]
|   +--rw order-id uint16
|   +--rw rule-name string
|   +--rw policy-name string
|   +--rw cfg-rule-conditions [rule-cnd-id]
|       +--rw rule-cnd-id uint32
|       | ...
|       +--rw eca-event-match* [rule-event-id]
|       | ..
|       +--rw eca-pkt-matches* [pkt-match-id]
|       | ....
|       +--rw eca-usr-context-matches* [usr-match-id]
|           +--rw user* [user-id]
|               +--rw user-id uint32
|               +--rw user-name string
|               +--rw user-type identityref
|                   +--(user-type)?
|                       +--:(tenant)
|                           +--rw tenant-id uint16
|                       +--:(vn-id)
|                           +--rw vn-id uint16
|           +--rw schedule* [schedule-name]
|               .....
|           +--rw target
|               +--rw protocol
|                   | ... (UDP, TCP, ICMP, ICMPv6, IP, IPv6)
|               +--rw transport-ports
|                   |   +--rw src-port inet:port-number
|                   |   +--rw dest-port intent:port-number
|               +--service [service-name]
|                   |...
|               +--rw application
|                   |...
|               +--rw device
|                   | ..

```

Figure 12

Schedule filters allow a time for the filter. Continuing our parental control filters for school, the schedule can be a list of weekly filters for Monday-Friday of the school week. The first filter (School-Monday) would have a start time of 8:00am GMT September 5, 2016 and an end time of 4:00pm GMT September 5, 2016. The schedule type would be weekly. The validity-until time would be December 20, 2016. The region impacted by this schedule would be AS20999 which is the service provider of the school's network.

```

+--rw pkt-eca-policy-cfg
|   +--rw pkt-eca-policy-set
|   +--rw pkt-eca-policy-set
|   |   ...
|   +--rw rules* [order-id rule-name]
|       +--rw order-id uint16
|       +--rw rule-name string
|       +--rw policy-name string
|       +--rw cfg-rule-conditions [rule-cnd-id]
|           +--rw rule-cnd-id uint32
|           +--rw eca-usr-context-matches* [usr-match-id]
|               +--rw schedule* [schedule-name]
|                   +--rw schedule-name
|                   +--rw schedule-type identityref /* one-time, weekly, 2 weeks,
monthly */
|                       +--rw start-type? yang:date-and-time
|                       +--rw end-type? yang:date-and-time
|                       +--rw validity-until yang:date-and-time /* valid until */
|                       +--rw region *[as-4byte]
|                       +--rw as-4byte uint32 /* region */

```

figure 13

The target for this service filtering is specified by protocols, applications, and devices. The figure below shows the filtering for a target protocol and port number or an application.

```

+--rw pkt-eca-policy-cfg
|   +--rw pkt-eca-policy-set
|   +--rw pkt-eca-policy-set
|   | ...
|   +--rw rules* [order-id rule-name]
|       +--rw order-id uint16
|       +--rw rule-name string
|       +--rw policy-name string
|       +--rw cfg-rule-conditions [rule-cnd-id]
|           +--rw rule-cnd-id uint32
|           +--rw eca-usr-context-matches* [usr-match-id]
|               +--rw target
|                   +--rw service* [svc-id svc-name]
|                       +--rw svc-id uint16
|                       +--rw svc-name string
|                       +--rw protocol-support
|                           +--rw TCP boolean
|                           +--rw UDP boolean
|                           +--rw ICMP boolean
|                           +--rw ICMPv6 boolean
|                           +--rw IP boolean
|                       +--rw src-port? inet:port-number
|                       +--rw dest-port? inet:port-number
|                   +--rw application* [app-name]
|                       +--rw app-name string
|                       +--rw app-id uint16
|                       +--rw app-category
|                           /* business, educational, internet */
|                       +--rw app-subcategory
|                           /* finance, email, game, social-net, web */
|                       +--rw app-data-transmission
|                           /* client-server, web-browser, p2p, network */
|                       +--rw app-risk-level
|                           /* exploitable, evasive, data-lost, malware-vehicle, tun
|                   +--rw device
|                       +--rw pc boolean
|                       +--rw mobile-phone boolean
|                       +--rw tablet boolean
|                       +--rw voip-phone boolean

```

Figure 14

4.4. ECA Packet Actions

The packet actions list includes ingress actions, egress actions, Qos actions that modify the packet, and security actions. The High level Yang that shows where the action fit is in figure 15, and the details

are shown in figure 16. The QoS actions per header is shown in figure 17.

```

module ietf-pkt-eca-policy
  +--rw pkt-eca-policy-cfg
  |   +--rw pkt-eca-policy-set
  |   |   +--rw policies* [policy-name]
  |   |   |   +--rw policy-name string
  |   |   |   +--rw vrf-name string
  |   |   |   +--rw address-family
  |   |   |   +--rw rule-list* [rule-name]
  |   |   |   |   +--rw rule-name
  |   |   |   |   +--rw rule-order-id uint16
  |   |   |   |   +--rw default-action-id integer
  |   |   |   |   +--rw default-resolution-strategy-id integer
  |   |   +--rw rules* [order-id rule-name]
  |   |   |   +--rw order-id uint16
  |   |   |   +--rw rule-name string
  |   |   |   +--rw cfg-rule-conditions [rule-cnd-id]
  |   |   |   |   ...
  |   |   |   +--rw cfg-rule-actions* [cfgr-action-id]
  |   |   |   |   +--rw cfgr-action-id
  |   |   |   |   +--rw eca-actions* [action-id]
  |   |   |   |   |   +--rw action-id uint32
  |   |   |   |   |   +--rw eca-ingress-actions
  |   |   |   |   |   |   ... (permit, deny, mirror)
  |   |   |   |   |   +--rw eca-fwd-actions*
  |   |   |   |   |   |   ... (invoke, tunnel encap, fwd)
  |   |   |   |   |   +--rw eca-egress-actions*
  |   |   |   |   |   |   ()
  |   |   |   |   |   +--rw eca-qos-actions*
  |   |   |   |   |   |   ...
  |   |   |   |   +--rw eca-security-actions*

```

Figure 15

This figure shows the details for each action section (ingress, egress, qos, and security).

```

|      +--rw eca-ingress-actions
|      |   +--rw num-fwd-actions
|      |   +--rw fwd-actions
|      |   |   +--rw permit boolean
|      |   |   +--rw mirror boolean
|      |   |   +--rw interface-fwd ip:interface-ref
|      |   |   +--uses i2rs:rib-nexthop
|      |   |   +--uses ip-next-fwd;
|      +--rw eca-egress-actions
|      |   +--rw packet-rate uint32
|      |   +--rw byte-rate uint32
|      |   +--rw tunnel-encap boolean
|      |   +--rw exit-fwdding boolean
|      |   +--rw interface-egress ip:interface-ref
|      |   +--uses i2rs:rib-nexthop
|      |   +--uses ip-next-fwd;
|      +--rw eca-qos-actions
|      |   ... (see figure x below )
|      +--rw eca-security
|
|      |   +--rw security-action-type identityref
|      |   +--(security-action-type)?
|      |   +--:(content-security-action) ANYXML
|      |   |   ...
|      |   +--:(attack-mitigation-type) ANYXML
|      |   |   ..
|      |   +--:(single-packet-type) ANYXML

```

figure 16 - forwarding

> The QOS actions modify the headers are shown below.

```

|      +--rw ecq-qos-actions
|      |   +--rw cnt-actions uint8 /* modifying actions */
|      |   +--rw mod-actions
|      |   |   +--case interface-actions
|      |   |   |   ..
|      |   |   +--case L1-action
|      |   |   |   ..
|      |   |   +--case L2-action
|      |   |   |   ..
|      |   |   +--case L3-action
|      |   |   |   ..
|      |   |   +--case L4-action
|      |   |   |   ..
|      |   |   +--case service-action
|      |   |   |   ..

```

Figure 17

4.5. Policy Conflict Resolution strategies

Some policies within the filter-base policy will conflict. For example, a global strategy may conflict with a local node strategy. This portion of the filter-based data model provides this support.

```

|      +--rw pc-resolution-strategies* [strategy-id]
|      |   +--rw strategy-id integer
|      |   +--rw pc-resolution-supported boolean
|      |   +--rw filter-strategy identityref
|      |   |   .. FMR, ADTP, Longest-match
|      |   +--rw global-strategy identityref
|      |   +--rw mandatory-strategy identityref
|      |   +--rw local-strategy identityref
|      |   +--rw resolution-fcn uint32
|      |   +--rw resolution-value uint32
|      |   +--rw resolution-info string
|      |   +--rw associated-ext-data*
|      |   |   +--rw ext-data-id integer

```

Figure 18

4.6. External Data

External data may be used to set the policy.

```

|         +---rw cfg-external-data* [cfg-ext-data-id]
|         |         +---rw cfg-ext-data-id integer
|         |         +---rw data-type integer
|         |         +---rw priority uint64
|         |         +---rw external-data-forms anyxml /* mount point */

```

Figure 19

5. i2rs-eca-policy Yang module

```

<CODE BEGINS> file "ietf-pkt-eca-policy@2016-02-09.yang"
module ietf-pkt-eca-policy {
  namespace "urn:ietf:params:xml:ns:yang:ietf-pkt-eca-policy";
  // replace with iana namespace when assigned
  prefix "pkt-eca-policy";

  import ietf-routing {
    prefix "rt";
  }
  import ietf-interfaces {
    prefix "if";
  }
  import ietf-inet-types {
    prefix inet;
    //rfc6991
  }

  import ietf-i2rs-rib {
    prefix "i2rs-rib";
  }

  // meta
  organization "IETF I2RS WG";

  contact
    "email: shares@ndzh.com
     email: russ.white@riw.com
     email: linda.dunbar@huawei.com
     email: bill.wu@huawei.com";

  description
    "This module describes a basic network policy
     model with filter per layer.";

  revision "2016-06-26" {
    description "sec ond revision";
    reference "draft-ietf-i2rs-pkt-eca-policy-dm-03";
  }

```

```
    }

// interfaces - no identity matches

// L1 header match identities
    identity l1-header-match-type {
        description
            " L1 header type for match ";
    }

identity l1-hdr-sonet-type {
    base l1-header-match-type;
    description
        " L1 header sonet match ";
}

identity l1-hdr-OTN-type {
    base l1-header-match-type;
    description
        " L1 header OTN match ";
}

    identity l1-hdr-dwdm-type {
        base l1-header-match-type;
        description
            " L1 header DWDM match ";
    }

// L2 header match identities
identity l2-header-match-type {
    description
        " L2 header type for match ";
}

identity l2-802-1Q {
    base l2-header-match-type;
    description
        " L2 header type for 802.1Q match ";
}

identity l2-802-11 {
    base l2-header-match-type;
    description
        " L2 header type for 802.11 match ";
}
```



```
    identity l2-802-15 {
base l2-header-match-type;
    description
    " l2 header type for 802.15 match ";
    }

    identity l2-NVGRE {
base l2-header-match-type;
description
    " l2 header type for NVGRE match ";
    }
    identity l2-mpls {
        base l2-header-match-type;
        description
    " l2 header type for MPLS match ";
    }

    identity l2-VXLAN {
base l2-header-match-type;
    description
    " l2 header type for VXLAN match ";
    }

    // L3 header match identities
    identity l3-header-match-type {
description
    " l3 header type for match ";
    }

    identity l3-ipv4-hdr {
        base l3-header-match-type;
        description
    " l3 header type for IPv4 match ";
    }

    identity l3-ipv6-hdr {
        base l3-header-match-type;
        description
    " l3 header type for IPv6 match ";
    }

    identity l3-gre-tunnel {
        base l3-header-match-type;
        description "l3 header r
        type for GRE tunnel match ";
    }
}
```

```
identity l3-icmp-header {
    base l3-header-match-type;
    description "L3 header match for ICMP";
}

identity l3-ipsec-ah-header {
    base l3-header-match-type;
    description "AH IPSEC header ";
}

identity l3-ipsec-esp-header {
    base l3-header-match-type;
    description "AH IPSEC header ";
}

// L4 header match identities

identity l4-header-match-type {
    description "L4 header
match types. (TCP, UDP,
SCTP, UDPLite, etc. )";
}

identity l4-tcp-header {
    base l4-header-match-type;
    description "L4 header for TCP";
}

identity l4-udp-header {
    base l4-header-match-type;
    description "L4 header match for UDP";
}

identity l4-udplite {
    base l4-header-match-type;
    description "L4 header match for
    UDP lite";
}

identity l4-sctp-header {
    base l4-header-match-type;
    description "L4 header match for SCTP";
}

// Service header identities

identity service-header-match-type {
```

```
    description "service header
    match types: service function path
    (sf-path)), SF-chain, sf-discovery,
    and others (added here)";
  }

  identity sf-chain-meta-match {
    base service-header-match-type;
    description "service header match for
    meta-match header";
  }

  identity sf-path-meta-match {
    base service-header-match-type;
    description "service header match for
    path-match header";
  }

  identity rule-status-type {
  description "status
    values for rule: invalid (0),
    valid (1), valid and installed (2)";
  }

  identity rule-status-invalid {
  base rule-status-type;
    description "invalid rule status.";
  }

  identity rule-status-valid {
    base rule-status-type;
    description "This status indicates
    a valid rule.";
  }

  identity rule-status-valid-installed {
    base rule-status-type;
    description "This status indicates
    an installed rule.";
  }
  identity rule-status-valid-inactive {
    base rule-status-type;
    description "This status indicates
    a valid ruled that is not installed.";
  }

  identity rule-cr-type {
```

```
description "status
  values for rule: FMR (0), ADTP (1),
  Longest-match (2)";
}

identity rule-cr-FMR {
  base rule-cr-type;
  description "first match resolution.";
}

identity rule-cr-ADTP {
  base rule-cr-type;
  description "ADTP resolution.";
}

identity rule-cr-longest {
  base rule-cr-type;
  description "longest match resolution.";
}

grouping interface-match {
  leaf match-if-name {
    type if:interface-ref;
    description "match on interface name";
  }
  description "interface
  has name, description, type, enabled
  as potential matches";
}

grouping interface-actions {
  description
  "interface action up/down and
  enable/disable";
  leaf interface-up {
    type boolean;
    description
    "action to put interface up";
  }
  leaf interface-down {
    type boolean;
    description
    "action to put interface down";
  }
  leaf interface-enable {
    type boolean;
  }
}
```

```
        description
        "action to enable interface";
    }
    leaf interface-disable {
type boolean;
        description
        "action to disable interface";
    }
}

grouping L1-header-match {
    choice l1-header-match-type {
        case l1-hdr-sonet-type {
            // sonet matches
        }
        case L1-hdr-OTN-type {
            // OTN matches
        }
        case L1-hdr-dwdm-type {
            // DWDM matches
        }
    }
    description
        "The Layer 1 header match choices";
}
description
    "The Layer 1 header match includes
    any reference to L1 technology";
}

grouping L1-header-actions {
    leaf l1-hdr-sonet-act {
        type uint8;
        description "sonet actions";
    }
    leaf l1-hdr-OTN-act {
        type uint8;
        description "OTN actions";
    }
    leaf l1-hdr-dwdm-act {
        type uint8;
        description "DWDM actions";
    }
    description "L1 header match
    types";
}

grouping L2-802-1Q-header {
```

```
description
  "This is short-term 802.1 header
  match which will be replaced
  by reference to IEEE yang when
  it arrives. Qtag 1 is 802.1Q
  Qtag2 is 802.1AD";

  leaf vlan-present {
    type boolean;
    description " Include VLAN in header";
  }
  leaf qtag1-present {
    type boolean;
    description " This flag value indicates
    inclusion of one 802.1Q tag in header";
  }
  leaf qtag2-present {
    type boolean;
    description "This flag indicates the
    inclusion of second 802.1Q tag in header";
  }

  leaf dest-mac {
    type uint64; //change to uint48
    description "IEEE destination MAC value
    from the header";
  }
  leaf src-mac {
    type uint64; //change to uint48
    description "IEEE source MAC
    from the header";
  }

  leaf vlan-tag {
    type uint16;
    description "IEEE VLAN Tag
    from the header";
  }
  leaf qtag1 {
    type uint32;
    description "Qtag1 value
    from the header";
  }
  leaf qtag2 {
    type uint32;
    description "Qtag1 value
    from the header";
  }
}
```

```
        leaf L2-ethertype {
            type uint16;
            description "Ether type
                        from the header";
        }
    }

    grouping L2-VXLAN-header {
        container vxlan-header {
            uses i2rs-rib:ipv4-header;
            leaf vxlan-network-id {
                type uint32;
                description "VLAN network id";
            }
            description " choices for
                        L2-VLAN header matches.
                        Outer-header only.
                        Need to fix inner header. ";
        }
        description
            "This VXLAN header may
            be replaced by actual VXLAN yang
            module reference";
    }

    grouping L2-NVGRE-header {

        container nvgre-header {
            uses L2-802-1Q-header;
            uses i2rs-rib:ipv4-header;
            leaf gre-version {
                type uint8;
                description "L2-NVGRE GRE version";
            }
            leaf gre-proto {
                type uint16;
                description "L2-NVGRE protocol value";
            }
            leaf virtual-subnet-id {
                type uint32;
                description "L2-NVGRE subnet id value";
            }
        }
        leaf flow-id {
            type uint16;
            description "L2-NVGRE Flow id value";
        }
        // uses L2-802-1Q-header;
```

```
    description
      "This NVGRE header may
      be replaced by actual NVGRE yang
      module reference";
  }
  description "Grouping for
    L2 NVGRE header.";
}

grouping L2-header-match {
  choice l2-header-match-type {
    case l2-802-1Q {
      uses L2-802-1Q-header;
    }
    case l2-802-11 {
      // matches for 802.11 headers
    }
    case l2-802-15 {
      // matches for 802.1 Ethernet
    }
    case l2-NVGRE {
      // matches for NVGRE
      uses L2-NVGRE-header;
    }
    case l2-VXLAN-header {
      uses L2-VXLAN-header;
    }
    case l2-mpls-header {
      uses i2rs-rib:mpls-header;
    }
    description "Choice of L2
      headers for L2 match";
  }
  description
    " The layer 2 header match includes
    any reference to L2 technology";
}

grouping L2-NVGRE-mod-acts {
  // actions for NVGRE
  leaf set-vsidi {
    type boolean;
    description
      "Boolean flag to set VSID in packet";
  }
  leaf set-flowid {
```



```
        type boolean;
        description
            "Boolean flag to set VSID in packet";
    }
    leaf vsi {
        type uint32;
        description
            "VSID value to set in packet";
    }
    leaf flow-id {
        type uint16;
        description
            "flow-id value to set in packet";
    }
    description "L2-NVRE Actions";
}

grouping L2-VXLAN-mod-acts {
    leaf set-network-id {
        type boolean;
        description
            "flag to set network id in packet";
    }
    leaf network-id {
        type uint32;
        description
            "network id value to set in packet";
    }
    description "VXLAN header
        modification actions.";
}

grouping L2-mpls-mod-acts {
    leaf pop {
        type boolean;
        description
            "Boolean flag to pop mpls header";
    }
    leaf push {
        type boolean;
        description
            "Boolean flag to push value into
            mpls header";
    }
    leaf mpls-label {
        type uint32;
        description
            "mpls label to push in header";
    }
}
```

```
    }
    description "MPLS modify
        header actions";
}

grouping l2-header-mod-actions {
    leaf l2-802-1Q {
        type uint8;
        description "actions for 802.1Q";
    }
    leaf l2-802-11 {
        type uint8;
        description "actions for 802.11";
    }
    leaf l2-802-15 {
        type uint8;
        description "actions for 802.15";
    }

    uses L2-NVGRE-mod-acts;
uses L2-VXLAN-mod-acts;
    uses L2-mpls-mod-acts;

    description
        " The layer 2 header match includes
        any reference to L2 technology";
}

grouping L3-header-match {

    choice L3-header-match-type {
        case l3-ipv4-hdr {
            uses i2rs-rib:ipv4-header;
        }
        case l3-ipv6-hdr {
            uses i2rs-rib:ipv6-header;
        }
        case L3-gre-tunnel {
            uses i2rs-rib:gre-header;
        }
        description "match for L3
            headers for IPv4, IPv6,
            and GRE tunnels";
    }
    description "match for L3 headers";
}
```

```
grouping ipv4-encapsulate-gre {
  leaf encapsulate {
    type boolean;
    description "flag to encapsulate headers";
  }
  leaf ipv4-dest-address {
    type inet:ipv4-address;
    description "Destination Address for GRE header";
  }
  leaf ipv4-source-address {
    type inet:ipv4-address;
    description "Source Address for GRE header";
  }
  description "encapsulation actions for IPv4 headers";
}

grouping L3-header-actions {
  choice l3-header-act-type {
    case l3-ipv4-hdr {
      leaf set-ttl {
        type boolean;
        description "flag to set TTL";
      }
      leaf set-dscp {
        type boolean;
        description "flag to set DSCP";
      }
      leaf ttl-value {
        type uint8;
        description "TTL value to set";
      }
      leaf dscp-val {
        type uint8;
        description "dscp value to set";
      }
    }
    case l3-ipv6-hdr {
      leaf set-next-header {
        type boolean;
        description
          "flag to set next routing
           header in IPv6 header";
      }
      leaf set-traffic-class {
        type boolean;
        description
          "flag to set traffic class
           in IPv6 header";
      }
    }
  }
}
```

```

    }
    leaf set-flow-label {
        type boolean;
        description
            "flag to set flow label
             in IPv6 header";
    }
    leaf set-hop-limit {
        type boolean;
        description "flag
            to set hop limit in
            L3 packet";
    }
    leaf ipv6-next-header {
        type uint8;
        description "value to
            set in next IPv6 header";
    }
    leaf ipv6-traffic-class {
        type uint8;
        description "value to set
            in traffic class";
    }

    }
    leaf ipv6-flow-label {
        type uint16;
        description "value to set
            in IPOv6 flow label";
    }
    leaf ipv6-hop-limit {
        type uint8;
        description "value to set
            in hop count";
    }
}

case L3-gre-tunnel {
    leaf decapsulate {
        type boolean;
        description "flag to
            decapsulate GRE packet";
    }
    description "GRE tunnel
        actions" ;
}
description "actions that can
    be performed on L3 header";
}

```

```
    description "actions to
    be performed on L3 header";
}

grouping tcp-header-match {
    leaf tcp-src-port {
        type uint16;
        description "source port match value";
    }
    leaf tcp-dst-port {
        type uint16;
        description "dest port value
        to match";
    }
    leaf sequence-number {
        type uint32;
        description "sequence number
        value to match";
    }
    leaf ack-number {
        type uint32;
        description "action value to
        match";
    }
    description "match for TCP
    header";
}

grouping tcp-header-action {
    leaf set-tcp-src-port {
        type boolean;
        description "flag to set
        source port value";
    }
    leaf set-tcp-dst-port {
        type boolean;
        description "flag to set source port value";
    }

    leaf tcp-s-port {
        type uint16;
        description "source port match value";
    }
    leaf tcp-d-port {
        type uint16;
        description "dest port value
        to match";
    }
}
```

```
    }
    leaf seq-num {
        type uint32;
        description "sequence number
            value to match";
    }
    leaf ack-num {
        type uint32;
        description "action value to
            match";
    }
    description "Actions to
        modify TCP header";
}

grouping udp-header-match {
    leaf udp-src-port {
        type uint16;
        description "UDP source
            port match value";
    }
    leaf udp-dst-port {
        type uint16;
        description "UDP Destination
            port match value";
    }
    description "match values for
        UDP header";
}

grouping udp-header-action {
    leaf set-udp-src-port {
        type boolean;
        description "flag to set
            UDP source port match value";
    }
    leaf set-udp-dst-port {
        type boolean;
        description
            "flag to set UDP destination port match value";
    }
    leaf udp-s-port {
        type uint16;
        description "UDP source
            port match value";
    }
    leaf udp-d-port {
```

```
        type uint16;
        description "UDP Destination
            port match value";
    }

    description "actions to set
        values in UDP header";
}

grouping sctp-chunk {
    leaf chunk-type {
        type uint8;
        description "sctp chunk type value";
    }
    leaf chunk-flag {
        type uint8;
        description "sctp chunk type
            flag value";
    }

    leaf chunk-length {
        type uint16;
        description "sctp chunk length";
    }

    leaf chunk-data-byte-zero {
        type uint32;
        description "byte zero of
            stcp chunk data";
    }
    description "sctp chunk
        header match fields";
}

grouping sctp-header-match {
    uses sctp-chunk;
    leaf stcp-src-port {
        type uint16;
        description "sctp header match
            source port value";
    }
    leaf sctp-dst-port {
        type uint16;
        description "sctp header match
            destination port value";
    }
    leaf sctp-verify-tag {
        type uint32;
    }
}
```

```
        description "sctp header match
            verification tag value";
    }
    description "SCTP header
        match values";
}

grouping sctp-header-action {
    leaf set-stcp-src-port {
        type boolean;
        description "set source port in sctp header";
    }
    leaf set-stcp-dst-port {
        type boolean;
        description "set destination port in sctp header";
    }
    leaf set-stcp-chunk1 {
        type boolean;
        description "set chunk value in sctp header";
    }
    leaf chunk-type-value {
        type uint8;
        description "sctp chunk type value";
    }
    leaf chunk-flag-value {
        type uint8;
        description "sctp chunk type
            flag value";
    }
}

    leaf chunk-len {
        type uint16;
        description "sctp chunk length";
    }

    leaf chunk-data-bzero {
        type uint32;
        description "byte zero of
            stcp chunk data";
    }
    description "sctp qos actions";
}

grouping L4-header-match {
    choice l4-header-match-type {
        case l4-tcp-header {
            uses tcp-header-match;
        }
    }
}
```



```
    }
    case l4-udp-header {
        uses udp-header-match;
    }
    case l4-sctp {
        uses sctp-header-match;
    }
    description "L4 match
header choices";
}
description "L4 header
match type";
}

grouping L4-header-actions {
    uses tcp-header-action;
    uses udp-header-action;
    uses sctp-header-action;
    description "L4 header matches";
}

grouping service-header-match {
    choice service-header-match-type {
        case sf-chain-meta-match {
            description "uses
sfc-sfc:service-function-chain-grouping:
+ sfc-sfc:service-function-chain";
        }
        case sf-path-meta-match {
            description "uses
sfc-spf:service-function-paths:
+ sfc-spf:service-function-path";
        }
    }
    description "SFC header match
choices";
}
description "SFC header and path
matches";
}

grouping sfc-header-actions {
    choice service-header-match-type {
        case sf-chain-meta-match {
            leaf set-chain {
                type boolean;
                description "flag to set
chain in sfc. Should
```

```

        be amended to use SFC service
        chain matching.
        uses sfc-sfc:service-function-chain-grouping:
        + sfc-sfc:service-function-chain";
    }
}
case sf-path-meta-match {
    leaf set-path {
        type boolean;
        description "flag to set path in
        sfc header. Amend to use sfc-spf
        function headers. Uses
        sfc-spf:service-function-paths:
        + sfc-spf:service-function-path.";
    }
}
description "choices in SFC for
chain match and path match.";
}
description "modify action for
SFC header.";
}

grouping rule_status {
    leaf rule-status {
        type string;
        description "status information
        free form string.";
    }
    leaf rule-inactive-reason {
        type string;
        description "description of
        why rule is inactive";
    }
    leaf rule-install-reason {
        type string;
        description "response on rule installed";
    }
    leaf rule-installer {
        type string;
        description "client id of installer";
    }
    leaf refcnt {
        type uint16;
        description "reference count on rule. ";
    }
}
description

```

```
        "rule operational status";
    }

// group status
grouping groups-status {
    list group_opstate {
        key "grp-name";
        leaf grp-name {
            type string;
            description "eca group name";
        }
        leaf rules-installed {
            type uint32;
            description "rules in
                group installed";
        }
        list rules_status {
            key "rule-name";
            leaf rule-name {
                type string;
                description "name of rule ";
            }
            leaf rule-order {
                type uint32;
                description "rule-order";
            }
            description "rules per
                group";
        }
        description "group operational
            status";
    }
    description "group to rules
        list";
}

// links between rule to group
grouping rule-group-link {
    list rule-group {
        key rule-name;
        leaf rule-name {
            type string;
            description "rule name";
        }
        leaf group-name {
            type string;
            description "group name";
        }
    }
}
```

```
    }
    description "link between
        group and link";
    }
    description "rule-name to
        group link";
    }

// rule status by name
grouping rules_opstate {
    list rules_status {
        key "rule-order rule-name";
        leaf rule-order {
            type uint32;
            description "order of rules";
        }
        leaf rule-name {
            type string;
            description "rule name";
        }
        uses rule_status;
        description "eca rule list";
    }
    description "rules
        operational state";
}

// rule statistics by name and order
grouping rules_opstats {
    list rule-stat {
        key "rule-order rule-name";
        leaf rule-order {
            type uint32;
            description "order of rules";
        }
        leaf rule-name {
            type string;
            description "name of rule";
        }
        leaf pkts-matched {
            type uint64;
            description "number of
                packets that matched filter";
        }
        leaf pkts-modified {
            type uint64;
            description "number of
                packets that filter caused
            ";
        }
    }
}
```

```
        to be modified";
    }
    leaf pkts-dropped {
        type uint64;
        description "number of
        packets that filter caused
        to be modified";
    }
    leaf bytes-dropped {
        type uint64;
        description "number of
        packets that filter caused
        to be modified";
    }
    leaf pkts-forwarded {
        type uint64;
        description "number of
        packets that filter caused
        to be forwarded.";
    }
    leaf bytes-forwarded {
        type uint64;
        description "number of
        packets that filter caused
        to be forwarded.";
    }

    description "list of
    operational statistics for each
    rule.";
}
description "statistics
on packet filter matches, and
based on matches on many were
modified and/or forwarded";
}

grouping packet-size-match {
    leaf l1-size-match {
        type uint32;
        description "L1 packet match size.";
    }
    leaf l2-size-match {
        type uint32;
        description "L2 packet match size.";
    }
}
```

```
    leaf l3-size-match {
        type uint32;
        description "L3 packet match size.";
    }
    leaf l4-size-match {
        type uint32;
        description "L4 packet match size.";
    }
    leaf service-meta-size {
        type uint32;
        description "service meta info match size.";
    }
    leaf service-meta-payload {
        type uint32;
        description "service meta-play match size";
    }
description "packet size by layer
only non-zero values are matched";
}

    grouping time-day-match {
leaf hour {
    type uint8;
    description "hour
of day in 24 hours.
(add range)";
}
leaf minute {
    type uint8;
    description
"minute in day.";
}
leaf second {
    type uint8;
    description
"second in day.";
}

description "matches for
time of day.";

}

    grouping user-event-match {
leaf user-name {
    type string;
```

```
        description "name of user
            event";
    }
    leaf match-string {
        type string;
        description "user match
            string";
    }

    description "matches for
        time of day.";
}

grouping eca-event-matches {
    uses time-day-match;
    uses user-event-match;
    description "matches for events
        which include:
            time of day, and
            user specified matches.";
}

grouping eca-pkt-matches {
    uses interface-match;
    uses L1-header-match;
    uses L2-header-match;
    uses L3-header-match;
    uses L4-header-match;
    uses service-header-match;
    uses packet-size-match;
    description "ECA matches";
}

grouping user-status-matches {
    leaf user {
        type string;
        description "user";
    }
    leaf region {
        type string;
        description "region";
    }
    leaf state {
        type string;
        description "state";
    }
}
```

```
    }

    leaf user-status {
      type string;
      description "status of user";
    }

    description "user status
matches - region,
target, location";
  }

  grouping eca-condition-matches {
    uses eca-pkt-matches;
    uses user-status-matches;
    description "pkt
      and user status matches";
  }

  grouping eca-qos-actions {
    leaf cnt-actions {
      type uint32;
      description "count of ECA actions";
    }
    list qos-actions {
      key "action-id";
      leaf action-id {
        type uint32;
        description "action id";
      }
      uses interface-actions;
      uses L1-header-actions;
      uses L2-header-mod-actions;
      uses L3-header-actions;
      uses L4-header-actions;

      description "ECA set or change
        packet Actions. Actions may be
        added here for interface,
        L1, L2, L3, L4 nad service forwarding
        headers.";
    }
    description "eca- qos actions";
  }

  grouping ip-next-fwd {
    leaf rib-name {
      type string;
    }
  }
```



```
        description "name of RIB";
    }
    leaf next-hop-name {
        type string;
        description "name of next hop";
    }
    description "ECA set or change
        packet Actions";
}

grouping eca-ingress-actions {
    leaf permit {
        type boolean;
        description "permit ingress
            traffic. False
                means to deny.";
    }
    leaf mirror {
        type boolean;
        description "copy bytes
            ingressed to mirror port";
    }
    description "ingress eca match";
}

grouping eca-fwd-actions {
    leaf interface-fwd {
        type if:interface-ref;
        description "name of interface to forward on";
    }
}
uses i2rs-rib:nexthop;
uses ip-next-fwd;
leaf drop-packet {
    type boolean;
    description "drop packet flag";
}
description "ECA forwarding actions";
}

grouping eca-security-actions {
    leaf actions-exist {
        type boolean;
        description "existence of
            eca security actions";
    }
}
description "content actions
    for security. Needs more
        description.";
```

```
}

grouping eca-egress-actions {
  leaf packet-rate {
    type uint32;
    description "maximum packet-rate";
  }
  leaf byte-rate {
    type uint64;
    description "maximum byte-rate ";
  }
  description "packet security actions";
}

grouping policy-conflict-resolution {
  list resolution-strategy {
    key "strategy-id";
    leaf strategy-id {
      type uint32;
      description "Id for strategy";
    }
    leaf strategy-name {
      type string;
      description "name of strategy";
    }
  }
  leaf filter-strategy {
    type string;
    description "type of resolution";
  }

  leaf global-strategy {
    type boolean;
    description "global strategy";
  }
  leaf mandatory-strategy {
    type boolean;
    description "required strategy";
  }
  leaf local-strategy {
    type boolean;
    description "local strategy";
  }
  leaf resolution-fcn {
    type uint64;
    description "resolution function id ";
  }
  leaf resolution-value {
```

```
    type uint64;
    description "resolution value";
  }
  leaf resolution-info {
    type string;
    description "resolution info";
  }
  list associate-ext-data {
    key "ext-data-id";
    leaf ext-data-id {
      type uint64;
      description "ID of external data";
    }
    leaf ext-data {
      type string;
      description "external data";
    }
    description "linked external data";
  }
  description "list of strategies";
}
description "policy conflict
  resolution strategies";
}
```

```
grouping cfg-external-data {
  list cfg-ext-data {
    key "cfg-ext-data-id";
    leaf cfg-ext-data-id {
      type uint64;
      description "id for external data";
    }
    leaf data-type {
      type uint32;
      description "external data type ID";
    }
    leaf priority {
      type uint64;
      description "priority of data";
    }
    leaf other-data {
      type string;
      description "string
        external data";
    }
    description "external data";
  }
}
```

```
    description "external data list";
}

grouping pkt-eca-policy-set {
  list groups {
    key "group-name";
    leaf group-name {
      type string;
      description
        "name of group of rules";
    }
    leaf vrf-name {
      type string;
      description "VRF name";
    }
    uses rt:address-family;
    list group-rule-list {
      key "rule-name";
      leaf rule-name {
        type string;
        description "name of rule";
      }
      leaf rule-order-id {
        type uint16;
        description "rule-order-id";
      }
      description "rules per group";
    }
    description "pkt eca rule groups";
  }
  list eca-rules {
    key "order-id";
    ordered-by user;
    leaf order-id {
      type uint16;
      description "Number of order
        in ordered list (ascending)";
    }
    leaf eca-rule-name {
      type string;
      description "name of rule";
    }
    leaf installer {
      type string;
      description
        "Id of I2RS client
        that installs this rule.";
    }
  }
}
```

```
    }
    uses eca-event-matches;
    uses eca-ingress-actions;
    uses eca-qos-actions;
    uses eca-security-actions;
    uses eca-fwd-actions;
    uses eca-egress-actions;
    uses cfg-external-data;
    uses policy-conflict-resolution;

    description "ECA rules";
  } // end of rule
description "Policy sets.";
}

grouping pkt-eca-opstate {
  uses groups-status;
  uses rule-group-link;
  uses rules_opstate;
  uses rules_opstats;
  description "pkt eca policy
    op-state main";
}

container pkt-eca-policy-opstate {
  config "false";
  uses pkt-eca-opstate;
  description "operational state";
}
}
```

<CODE ENDS>

6. IANA Considerations

This draft requests IANA Assign a urn in the IETF yang module space for:

"urn:ietf:params:xml:ns:yang:ietf-pkt-eca-policy";

associated prefix "pkt-eca";

7. Security Considerations

These generic filters are used in the I2RS FB-RIBs to filter packets in a traffic stream, act to modify packets, and forward data packets. These I2RS filters operate dynamically at same level as currently deployed configured filter-based RIBs to filter, change, and forward traffic. The dynamic nature of this protocol requires that I2RS Filters track the installer of group information and rules.

This section will be augmented after a discussion with security experts.

8. Informative References

- [I-D.ietf-i2rs-rib-info-model]
Bahadur, N., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-09 (work in progress), July 2016.
- [I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-17 (work in progress), September 2016.
- [I-D.ietf-netmod-acl-model]
Bogdanovic, D., Koushik, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-08 (work in progress), July 2016.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3060] Moore, B., Ellessen, E., Strassner, J., and A. Westerinen, "Policy Core Information Model -- Version 1 Specification", RFC 3060, DOI 10.17487/RFC3060, February 2001, <<http://www.rfc-editor.org/info/rfc3060>>.
- [RFC3460] Moore, B., Ed., "Policy Core Information Model (PCIM) Extensions", RFC 3460, DOI 10.17487/RFC3460, January 2003, <<http://www.rfc-editor.org/info/rfc3460>>.
- [RFC3644] Snir, Y., Ramberg, Y., Strassner, J., Cohen, R., and B. Moore, "Policy Quality of Service (QoS) Information Model", RFC 3644, DOI 10.17487/RFC3644, November 2003, <<http://www.rfc-editor.org/info/rfc3644>>.

[RFC7921] Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", RFC 7921, DOI 10.17487/RFC7921, June 2016, <<http://www.rfc-editor.org/info/rfc7921>>.

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: bill.wu@huawei.com

Russ White
Ericsson

Email: russw@riw.us

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2017

M. Bjorklund, Ed.
Tail-f Systems
J. Schoenwaelder
Jacobs University
P. Shafer
K. Watsen
Juniper
R. Wilton
Cisco
October 27, 2016

A Revised Conceptual Model for YANG Datastores
draft-nmdsdt-netmod-revised-datastores-00

Abstract

Datastores are a fundamental concept binding the YANG data modeling language to protocols transporting data defined in YANG data models, such as NETCONF or RESTCONF. This document defines a revised conceptual model of datastores based on the experience gained with the initial simpler model and addressing requirements that were not well supported in the initial model.

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 30, 2017.

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. Background	3
3. Terminology	4
4. Original Model of Datastores	4
5. Revised Model of Datastores	6
5.1. The <intended> datastore	8
5.2. The <applied> datastore	8
5.2.1. Missing Resources	9
5.2.2. System-controlled Resources	9
5.3. The <operational-state> datastore	9
6. Implications	9
6.1. Implications on NETCONF	9
6.1.1. Migration Path	10
6.2. Implications on RESTCONF	10
6.3. Implications on YANG	11
6.4. Implications on Data Models	11
7. Data Model Design Guidelines	11
7.1. Auto-configured or Auto-negotiated Values	11
8. Data Model	12
9. IANA Considerations	14
10. Security Considerations	14
11. Acknowledgments	14
12. References	15
12.1. Normative References	15
12.2. Informative References	15
Appendix A. Example Data	16
Appendix B. Open Issues	19
Authors' Addresses	20

1. Introduction

This document provides a revised architectural framework for datastores as they are used by network management protocols such as NETCONF [RFC6241], RESTCONF [I-D.ietf-netconf-restconf] and the YANG [RFC7950] data modeling language. Datastores are a fundamental concept binding management data models to network management protocols and agreement on a common architectural model of datastores ensures that data models can be written in a network management

protocol agnostic way. This architectural framework identifies a set of conceptual datastores but it does not mandate that all network management protocols expose all these conceptual datastores. Furthermore, the architecture does not detail how data is encoded by network management protocols.

2. Background

NETCONF [RFC6241] provides the following definitions:

- o datastore: A conceptual place to store and access information. A datastore might be implemented, for example, using files, a database, flash memory locations, or combinations thereof.
- o configuration datastore: The datastore holding the complete set of configuration data that is required to get a device from its initial default state into a desired operational state.

YANG 1.1 [RFC7950] provides the following refinements when NETCONF is used with YANG (which is the usual case but note that NETCONF was defined before YANG did exist):

- o datastore: When modeled with YANG, a datastore is realized as an instantiated data tree.
- o configuration datastore: When modeled with YANG, a configuration datastore is realized as an instantiated data tree with configuration data.

RFC 6244 defined operational state data as follows:

- o Operational state data is a set of data that has been obtained by the system at runtime and influences the system's behavior similar to configuration data. In contrast to configuration data, operational state is transient and modified by interactions with internal components or other systems via specialized protocols.

Section 4.3.3 of RFC 6244 discusses operational state and among other things mentions the option to consider operational state as being stored in another datastore. Section 4.4 of this document then concludes that at the time of the writing, modeling state as a separate data tree is the recommended approach.

Implementation experience and requests from operators [I-D.ietf-netmod-opstate-reqs], [I-D.openconfig-netmod-opstate] indicate that the datastore model initially designed for NETCONF and refined by YANG needs to be extended. In particular, the notion of intended configuration and applied configuration has developed.

Furthermore, separating operational state data from configuration data in a separate branch in the data model has been found operationally complicated. The relationship between the branches is not machine readable and filter expressions operating on configuration data and on related operational state data are different.

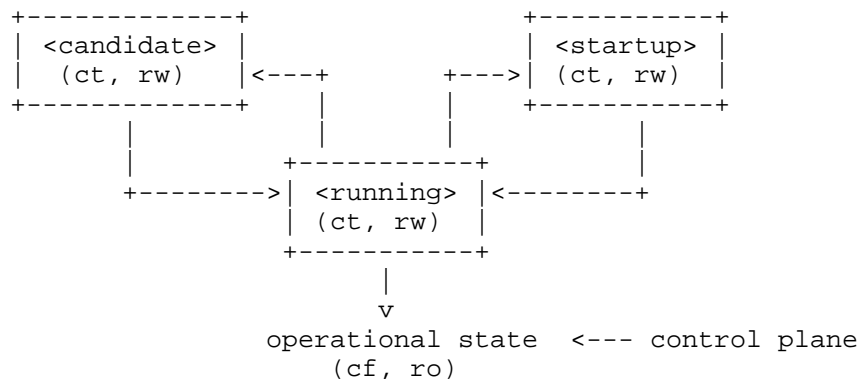
3. Terminology

This document defines the following terms:

- o configuration data: Data that determines how a device behaves. Configuration data can originate from different sources. In YANG 1.1, configuration data is the "config true" nodes.
- o static configuration data: Configuration data that is eventually persistent and used to get a device from its initial default state into its desired operational state.
- o dynamic configuration data: Configuration data that is obtained dynamically during the operation of a device through interaction with other systems and not persistent.
- o system configuration data: Configuration data that is supplied by the device itself.
- o data-model-defined configuration data: Configuration data that is not explicitly provided but for which a value defined in the data model is used. In YANG 1.1, such data can be defined with the "default" statement or in "description" statements.

4. Original Model of Datastores

The following drawing shows the original model of datastores as it is currently used by NETCONF [RFC6241]:



ct = config true; cf = config false
 rw = read-write; ro = read-only
 boxes denote datastores

Note that read-only (ro) and read-write (rw) is to be understood at a conceptual level. In NETCONF, for example, support for the <candidate> and <startup> datastores is optional and the <running> datastore does not have to be writable. Furthermore, the <startup> datastore can only be modified by copying <running> to <startup> in the standardized NETCONF datastore editing model. The RESTCONF protocol does not expose these differences and instead provides only a writable unified datastore, which hides whether edits are done through a <candidate> datastore or by directly modifying the <running> datastore or via some other implementation specific mechanism. RESTCONF also hides how configuration is made persistent. Note that implementations may also have additional datastores that can propagate changes to the <running> datastore. NETCONF explicitly mentions so called named datastores.

Some observations:

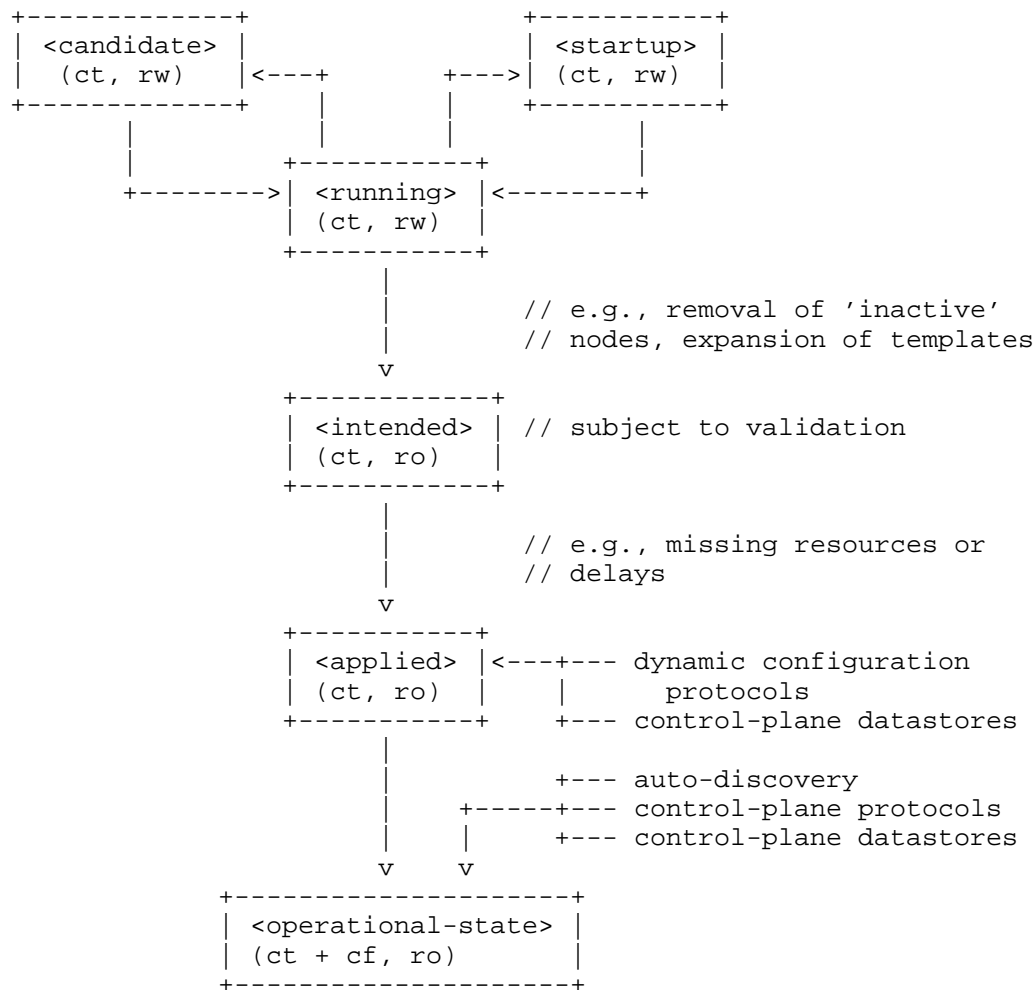
- o Operational state has not been defined as a datastore although there were proposals in the past to introduce an operational state datastore.
- o The NETCONF <get/> operation returns the content of the <running> configuration datastore together with the operational state. It is therefore necessary that config false data is in a different branch than the config true data if the operational state data can have a different lifetime compared to configuration data or if configuration data is not immediately or successfully applied.
- o Several implementations have proprietary mechanisms that allow clients to store inactive data in the <running> datastore; this

inactive data is only exposed to clients that indicate that they support the concept of inactive data; clients not indicating support for inactive data receive the content of the <running> datastore with the inactive data removed. Inactive data is conceptually removed during validation.

- o Some implementations have proprietary mechanisms that allow clients to define configuration templates in <running>. These templates are expanded automatically by the system, and the resulting configuration is applied internally.
- o Some operators have reported that it is essential for them to be able to retrieve the configuration that has actually been successfully applied, which may be a subset or a superset of the <running> configuration.

5. Revised Model of Datastores

Below is a new conceptual model of datastores extending the original model in order to reflect the experience gained with the original model.



ct = config true; cf = config false
 rw = read-write; ro = read-only
 boxes denote datastores

The model foresees control-plane datastores that are by definition not part of the persistent configuration of a device. In some contexts, these have been termed ephemeral datastores since the information is ephemeral, i.e., lost upon reboot. The control-plane datastores interact with the rest of the system through the <applied> or <operational-state> datastores, depending on the type of data they contain. Note that the ephemeral datastore discussed in I2RS documents maps to a control-plane datastore in the revised datastore model described here.

5.1. The <intended> datastore

The <intended> datastore is a read-only datastore that consists of config true nodes. It is tightly coupled to <running>. When data is written to <running>, the data that is to be validated is also conceptually written to <intended>. Validation is performed on the contents of <intended>.

On a traditional NETCONF implementation, <running> and <intended> are always the same.

Currently there are no standard mechanisms defined that affect <intended> so that it would have different contents than <running>, but this architecture allows for such mechanisms to be defined.

One example of such a mechanism is support for marking nodes as inactive in <running>. Inactive nodes are not copied to <intended>, and are thus not taken into account when validating the configuration.

Another example is support for templates. Templates are expanded when copied into <intended>, and the result is validated.

5.2. The <applied> datastore

The <applied> datastore is a read-only datastore that consists of config true nodes. It contains the currently active configuration on the device. This data can come from several sources; from <intended>, from dynamic configuration protocols (e.g., DHCP), or from control-plane datastores.

As data flows into the <applied> and <operational-state> datastores, it is conceptually marked with a metadata annotation ([RFC7952]) that indicates its origin. The "origin" metadata annotation is defined in Section 8. The values are YANG identities. The following identities are defined:

```
+-- origin
  +-- static
  +-- dynamic
  +-- data-model
  +-- system
```

These identities can be further refined, e.g., there might be an identity "dhcp" derived from "dynamic".

The <applied> datastore contains the subset of the instances in the <operational-state> datastore where the "origin" values are derived from or equal to "static" or "dynamic".

5.2.1. Missing Resources

Sometimes some parts of <intended> configuration refer to resources that are not present and hence parts of the <intended> configuration cannot be applied. A typical example is an interface configuration that refers to an interface that is not currently present. In such a situation, the interface configuration remains in <intended> but the interface configuration will not appear in <applied>.

5.2.2. System-controlled Resources

Sometimes resources are controlled by the device and such system controlled resources appear in (and disappear from) the <operational-state> dynamically. If a system controlled resource has matching configuration in <intended> when it appears, the system will try to apply the configuration, which causes the configuration to appear in <applied> eventually (if application of the configuration was successful).

5.3. The <operational-state> datastore

The <operational-state> datastore is a read-only datastore that consists of config true and config false nodes. In the original NETCONF model the operational state only had config false nodes. The reason for incorporating config true nodes here is to be able to expose all operational settings without having to replicate definitions in the data models.

The <operational-state> datastore contains all configura data actually used by the system, i.e., all applied configuration, system configuration and data-model-defined configuration. This data is marked with the "origin" metadata annotation. In addition, the <operational-state> datastore also contains state data.

In the <operational-state> datastore, semantic constraints defined in the data model are not applied. See Appendix B.

6. Implications

6.1. Implications on NETCONF

- o A mechanism is needed to announce support for <intended>, <applied>, and <operational-state>.

- o Support for `<intended>`, `<applied>`, and `<operational-state>` should be optional to implement.
- o For systems supporting `<intended>` or `<applied>` configuration datastores, the `<get-config/>` operation may be used to retrieve data stored in these new datastores.
- o A new operation should be added to retrieve the operational state data store (e.g., `<get-state/>`). An alternative is to define a new operation to retrieve data from any datastore (e.g., `<get-data>` with the name of the datastore as a parameter). In principle `<get-config/>` could work but it would be a confusing name.
- o The `<get/>` operation will be deprecated since it returns data from two datastores that may overlap in the revised datastore model.

6.1.1. Migration Path

A common approach in current data models is to have two separate trees `"/foo"` and `"/foo-state"`, where the former contains config true nodes, and the latter config false nodes. A data model that is designed for the revised architectural framework presented in this document will have a single tree `"/foo"` with a combination of config true and config false nodes.

A server that implements the `<operational-state>` datastore can implement a module of the old design. In this case, some instances are probably reported both in the `"/foo"` tree and in the `"/foo-state"` tree.

A server that does not implement the `<operational-state>` datastore can implement a module of the new design, but with limited functionality. Specifically, it may not be possible to retrieve all operationally used instances (e.g., dynamically configured or system-controlled). The same limitation applies to a client that does not implement the `<operational-state>` datastore, but talks to a server that implements it.

6.2. Implications on RESTCONF

- o The `{+restconf}/data` resource represents the combined configuration and state data resources that can be accessed by a client. This is effectively bundling `<running>` together with `<operational-state>`, much like the `<get/>` operation of NETCONF. This design should be deprecated.

- o A new query parameter is needed to indicate that data from <operational-state> is requested.

6.3. Implications on YANG

- o Some clarifications may be needed if this revised model is adopted. YANG currently describes validation in terms of the <running> configuration datastore while it really happens on the <intended> configuration datastore.

6.4. Implications on Data Models

- o Since the NETCONF <get/> operation returns the content of the <running> configuration datastore and the operational state together in one tree, data models were often forced to branch at the top-level into a config true branch and a structurally similar config false branch that replicated some of the config true nodes and added state nodes. With the revised datastore model this is not needed anymore since the different datastores handle the different lifetimes of data objects. Introducing this model together with the deprecation of the <get/> operation makes it possible to write simpler models.
- o There may be some differences in the value set of some nodes that are used for both configuration and state. At this point of time, these are considered to be rare cases that can be dealt with using different nodes for the configured and state values.
- o It is important to design data models with clear semantics that work equally well for instantiation in a configuration datastore and instantiation in the <operational-state> datastore.

7. Data Model Design Guidelines

7.1. Auto-configured or Auto-negotiated Values

Sometimes configuration leafs support special values that instruct the system to automatically configure a value. An example is an MTU that is configured to 'auto' to let the system determine a suitable MTU value. Another example is Ethernet auto-negotiation of link speed. In such a situation, it is recommended to model this as two separate leafs, one config true leaf for the input to the auto-negotiation process, and one config false leaf for the output from the process.

8. Data Model

```
<CODE BEGINS> file "ietf-yang-architecture@2016-10-13.yang"

module ietf-yang-architecture {
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-architecture";
  prefix arch;

  import ietf-yang-metadata {
    prefix md;
  }

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

  contact
    "WG Web:  <https://datatracker.ietf.org/wg/netmod/>

    WG List:  <mailto:netmod@ietf.org>

    Editor:   Martin Bjorklund
              <mailto:mbj@tail-f.com>";

  description
    "This YANG module defines an 'origin' metadata annotation,
    and a set of identities for the origin value.  The 'origin'
    metadata annotation is used to mark data in the applied
    and operational state datastores with information on where
    the data originated.

    Copyright (c) 2016 IETF Trust and the persons identified as
    authors of the code.  All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject to
    the license terms contained in, the Simplified BSD License set
    forth in Section 4.c of the IETF Trust's Legal Provisions
    Relating to IETF Documents
    (http://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC XXXX
    (http://www.rfc-editor.org/info/rfcxxxx); see the RFC itself
    for full legal notices.";

  revision 2016-10-13 {
    description
      "Initial revision.";
    reference
```

```
    "RFC XXXX: A Revised Conceptual Model for YANG Datastores";
}

/*
 * Identities
 */

identity origin {
    description
        "Abstract base identity for the origin annotation.";
}

identity static {
    base origin;
    description
        "Denotes data from static configuration (e.g., <intended>).";
}

identity dynamic {
    base origin;
    description
        "Denotes data from dynamic configuration protocols
        or dynamic datastores (e.g., DHCP).";
}

identity system {
    base origin;
    description
        "Denotes data created by the system independently of what
        has been configured.";
}

identity data-model {
    base origin;
    description
        "Denotes data that does not have an explicitly configured
        value, but has a default value in use.  Covers both simple
        defaults and complex defaults.";
}

/*
 * Metadata annotations
 */

md:annotation origin {
    type identityref {
        base origin;
    }
}
```

```
}  
  
}  
  
<CODE ENDS>
```

9. IANA Considerations

TBD

10. Security Considerations

This document discusses a conceptual model of datastores for network management using NETCONF/RESTCONF and YANG. It has no security impact on the Internet.

11. Acknowledgments

This document grew out of many discussions that took place since 2010. Several Internet-Drafts ([I-D.bjorklund-netmod-operational], [I-D.wilton-netmod-opstate-yang], [I-D.ietf-netmod-opstate-reqs], [I-D.kwatsen-netmod-opstate], [I-D.openconfig-netmod-opstate]) and [RFC6244] touched on some of the problems of the original datastore model. The following people were authors to these Internet-Drafts or otherwise actively involved in the discussions that led to this document:

- o Lou Berger, LabN Consulting, L.L.C., <lberger@labn.net>
- o Andy Bierman, YumaWorks, <andy@yumaworks.com>
- o Marcus Hines, Google, <hines@google.com>
- o Christian Hopps, Deutsche Telekom, <chopps@chopps.org>
- o Acee Lindem, Cisco Systems, <acee@cisco.com>
- o Ladislav Lhotka, CZ.NIC, <lhotka@nic.cz>
- o Thomas Nadeau, Brocade Networks, <tnadeau@lucidvision.com>
- o Anees Shaikh, Google, <aashaikh@google.com>
- o Rob Shakir, Google, <robjs@google.com>

Juergen Schoenwaelder was partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.

12. References

12.1. Normative References

- [I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-17 (work in progress), September 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>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<http://www.rfc-editor.org/info/rfc7950>>.
- [RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <<http://www.rfc-editor.org/info/rfc7952>>.

12.2. Informative References

- [I-D.bjorklund-netmod-operational]
Bjorklund, M. and L. Lhotka, "Operational Data in NETCONF and YANG", draft-bjorklund-netmod-operational-00 (work in progress), October 2012.
- [I-D.ietf-netmod-opstate-reqs]
Watsen, K. and T. Nadeau, "Terminology and Requirements for Enhanced Handling of Operational State", draft-ietf-netmod-opstate-reqs-04 (work in progress), January 2016.
- [I-D.kwatsen-netmod-opstate]
Watsen, K., Bierman, A., Bjorklund, M., and J. Schoenwaelder, "Operational State Enhancements for YANG, NETCONF, and RESTCONF", draft-kwatsen-netmod-opstate-02 (work in progress), February 2016.
- [I-D.openconfig-netmod-opstate]
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-01 (work in progress), July 2015.

- [I-D.wilton-netmod-opstate-yang]
Wilton, R., "With-config-state" Capability for NETCONF/
RESTCONF", draft-wilton-netmod-opstate-yang-02 (work in
progress), December 2015.
- [RFC6244] Shafer, P., "An Architecture for Network Management Using
NETCONF and YANG", RFC 6244, DOI 10.17487/RFC6244, June
2011, <<http://www.rfc-editor.org/info/rfc6244>>.

Appendix A. Example Data

In this example, the following fictional module is used:

```
module example-system {
  yang-version 1.1;
  namespace urn:example:system;
  prefix sys;

  import ietf-inet-types {
    prefix inet;
  }

  container system {
    leaf hostname {
      type string;
    }
  }

  list interface {
    key name;

    leaf name {
      type string;
    }

    container auto-negotiation {
      leaf enabled {
        type boolean;
        default true;
      }
      leaf speed {
        type uint32;
        units mbps;
        description
          "The advertised speed, in mbps.";
      }
    }
  }

  leaf speed {
```

```
        type uint32;
        units mbps;
        config false;
        description
            "The speed of the interface, in mbps.";
    }

    list address {
        key ip;

        leaf ip {
            type inet:ip-address;
        }
        leaf prefix-length {
            type uint8;
        }
    }
}
}
```

The operator has configured the host name and two interfaces, so the contents of <intended> is:

```
<system xmlns="urn:example:system">

  <hostname>foo</hostname>

  <interface>
    <name>eth0</name>
    <auto-negotiation>
      <speed>1000</speed>
    </auto-negotiation>
    <address>
      <ip>2001:db8::10</ip>
      <prefix-length>32</prefix-length>
    </address>
  </interface>

  <interface>
    <name>eth1</name>
    <address>
      <ip>2001:db8::20</ip>
      <prefix-length>32</prefix-length>
    </address>
  </interface>

</system>
```


The system has detected that the hardware for one of the configured interfaces ("eth1") is not yet present, so the configuration for that interface is not applied. Further, the system has received a host name and an additional IP address for "eth0" over DHCP. This is reflected in <applied>:

```
<system
  xmlns="urn:example:system"
  xmlns:arch="urn:ietf:params:xml:ns:yang:ietf-yang-architecture">

  <hostname arch:origin="arch:dynamic">bar</hostname>

  <interface arch:origin="arch:static">
    <name>eth0</name>
    <auto-negotiation>
      <speed>1000</speed>
    </auto-negotiation>
    <address>
      <ip>2001:db8::10</ip>
      <prefix-length>32</prefix-length>
    </address>
    <address arch:origin="arch:dynamic">
      <ip>2001:db8::1:100</ip>
      <prefix-length>32</prefix-length>
    </address>
  </interface>

</system>
```

In <operational-state>, all data from <applied> is present, in addition to a default value, a loopback interface automatically added by the system, and the result of the "speed" auto-negotiation:

```
<system
  xmlns="urn:example:system"
  xmlns:arch="urn:ietf:params:xml:ns:yang:ietf-yang-architecture">

  <hostname arch:origin="arch:dynamic">bar</hostname>

  <interface arch:origin="arch:static">
    <name>eth0</name>
    <auto-negotiation>
      <enabled arch:origin="arch:data-model">true</enabled>
      <speed>1000</speed>
    </auto-negotiation>
    <speed>100</speed>
    <address>
      <ip>2001:db8::10</ip>
      <prefix-length>32</prefix-length>
    </address>
    <address arch:origin="arch:dynamic">
      <ip>2001:db8::1:100</ip>
      <prefix-length>32</prefix-length>
    </address>
  </interface>

  <interface arch:origin="arch:system">
    <name>lo0</name>
    <address>
      <ip>::1</ip>
      <prefix-length>128</prefix-length>
    </address>
  </interface>

</system>
```

Appendix B. Open Issues

1. Do we need another DS `<active>` inbetween `<running>` and `<intended>`? This DS would allow a client to see all active nodes, including unexpanded templates.
2. How do we handle semantical constraints in `<operational-state>`? Are they just ignored? Do we need a new YANG statement to define if a "must" constraints applies to the `<operational-state>`?
3. Should it be possible to ask for `<applied>` in RESTCONF?
4. Better name for "static configuration"?
5. Better name for "intended"?

Authors' Addresses

Martin Bjorklund (editor)
Tail-f Systems

Email: mbj@tail-f.com

Juergen Schoenwaelder
Jacobs University

Email: j.schoenwaelder@jacobs-university.de

Phil Shafer
Juniper

Email: phil@juniper.net

Kent Watsen
Juniper

Email: kwatsen@juniper.net

Rob Wilton
Cisco

Email: rwilton@cisco.com

I2RS Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2018

Y. Zhuang
D. Shi
Huawei
R. Gu
China Mobile
H. Ananthakrishnan
Packet Design
July 3, 2017

A YANG Data Model for Fabric Topology in Data Center Network
draft-zhuang-i2rs-yang-dc-fabric-network-topology-04

Abstract

This document defines a YANG data model for fabric topology in Data Center Network.

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 4, 2018.

Copyright Notice

Copyright (c) 2017 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 an Acronyms	3
2.1. Terminology	3
2.2. Tree diagram	4
3. Model Overview	4
3.1. Topology Model structure	4
3.2. Fabric Topology Model	5
3.2.1. Fabric Topology	5
3.2.2. Fabric node extension	6
3.2.3. Fabric termination-point extension	7
4. Fabric YANG Module	8
5. Security Consideration	21
6. Acknowledgements	21
7. References	21
7.1. Normative References	21
7.2. Informative References	22
Appendix A. Non NMDA -state modules	22
Authors' Addresses	27

1. Introduction

Normally, a data center network is composed of single or multiple fabrics which are also known as PODs (a Point Of Delivery). These fabrics may be heterogeneous due to implementation of different technologies while DC network upgrading or enrolling new techniques and features. For example, Fabric A may use VXLAN while Fabric B may use VLAN within a DC network. Likewise, a legacy Fabric may use VXLAN while a new Fabric B implemented technique discussed in NVO3 WG such as GPE[I-D. draft-ietf-nvo3-vxlan-gpe] may be built due to DC expansion and upgrading. The configuration and management of such DC networks with heterogeneous fabrics will be sophisticated and complex.

Luckily, for a DC network, a fabric can be considered as an atomic structure to provide network services and management, as well as expand network capacity. From this point of view, the miscellaneous DC network management can be decomposed to task of managing each fabric respectively along with their connections, which can make the entire management much concentrated and flexible, also easy to expand.

With this purpose, this document defines a YANG data model for the Fabric-based Data center topology by using YANG [6020][7950]. To do

so, it augments the generic network and network topology data models defined in [I-D.ietf-i2rs-yang-network-topo] with information specific to Data Center fabric network.

This model defines the generic configuration and operational state for a fabric-based network topology, which can be extended by vendors with specific information. This model can then be used by a network controller to represent its view of the fabric topology that it controls and expose it to network administrators or applications for DC network management.

With the context of topology architecture defined in [I-D.ietf-i2rs-yang-network-topo] and [I.D. draft-ietf-i2rs-usecase-reqs-summary], this model can also be treated as an application of I2RS network topology model [I-D.ietf-i2rs-yang-network-topo] in the scenario of Data center network management. It can also act as a service topology when mapping network elements at fabric layer to elements to other topologies, such as L3 topology defined in [I.D. draft-ietf-i2rs-yang-l3-topology-01].

By using this fabric topology model, people can treat a fabric as an entity and focus on characteristics of fabrics (such as encapsulation type, gateway type, etc.) as well as their interconnections while putting the underlay topology aside. As such, clients can consume the topology information at fabric level, while no need to be aware of entire set of links and nodes in underlay networks. The configuration of a fabric topology can be made by a network administrator to the controller by adding physical devices and links of a fabric into a fabric network. Alternatively, the fabric topology can also learnt from the underlay network infrastructure.

2. Definitions and Acronyms

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

2.1. Terminology

DC Fabric: also known as POD, is a module of network, compute, storage, and application components that work together to deliver networking services. It is a repeatable design pattern, and its components maximize the modularity, scalability, and manageability of data centers.

2.2. Tree diagram

The following notations are used within the data tree and carry the meaning as below.

Each node is printed as:

`<status> <flags> <name> <opts> <type>`

`<status>` is one of:

- + for current
- x for deprecated
- o for obsolete

`<flags>` is one of:

- rw for configuration data
- ro for non-configuration data
- x for rpcs
- n for notifications

`<name>` is the name of the node

If the node is augmented into the tree from another module, its name is printed as `<prefix>:<name>`.

`<opts>` is one of:

- ? for an optional leaf or choice
- ! for a presence container
- * for a leaf-list or list
- [<keys>] for a list's keys

`<type>` is the name of the type for leafs and leaf-lists

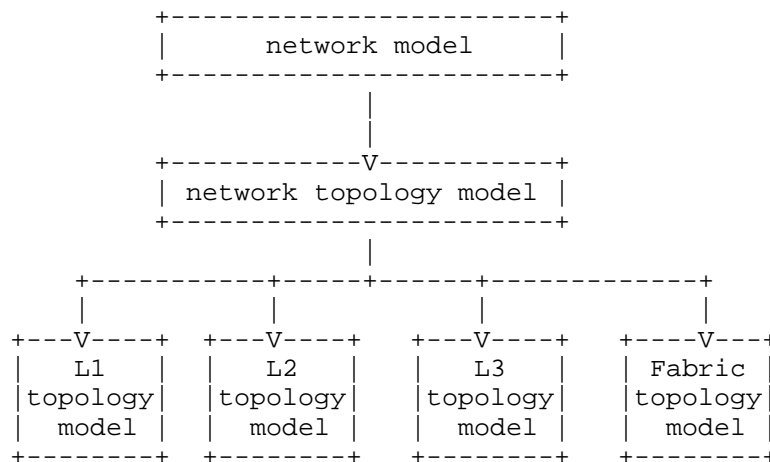
In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

3. Model Overview

This section provides an overview of the DC Fabric topology model and its relationship with other topology models.

3.1. Topology Model structure

The relationship of the DC fabric topology model and other topology models is shown in the following figure (dotted lines in the figure denote augmentations).



From the perspective of resource management and service provisioning for a Data Center network, the fabric topology model augments the basic network topology model with definitions and features specific to a DC fabric, to provide common configuration and operations for heterogeneous fabrics.

3.2. Fabric Topology Model

The fabric topology model module is designed to be generic and can be applied to data center fabrics built with different technologies, such as VLAN, VXLAN etc al. The main purpose of this module is to configure and manage fabrics and their connections. provide a fabric-based topology view for data center network applications.

3.2.1. Fabric Topology

In the fabric topology module, a fabric is modeled as a node in the network, while the fabric-based Data center network consists of a set of fabric nodes and their connections known as "fabric port". The following is the snatch of the definition to show the main structure of the model:


```

module: ietf-fabric-topology
augment /nw:networks/nw:network/nw:network-types:
  +--rw fabric-network!
augment /nw:networks/nw:network/nw:node:
  +--rw fabric-attribute
    +--rw name?          string
    +--rw type?          fabrictype:underlayer-network-type
    +--rw description?   string
    +--rw options
    +--...
augment /nw:networks/nw:network/nw:node/nt:termination-point:
  +--ro fport-attribute
    +--ro name?          string
    +--ro role?          fabric-port-role
    +--ro type?          fabric-port-type

```

The fabric topology module augments the generic ietf-network and ietf-network-topology modules as follows:

- o A new topology type "ietf-fabric-topology" is introduced and added under the "network-types" container of the ietf-network module.
- o Fabric is defined as a node under the network/node container. A new container of "fabric-attribute" is defined to carry attributes for a fabric network such as gateway mode, fabric types, involved device nodes and links etc al.
- o Termination points (in network topology module) are augmented with fabric port attributes defined in a container. The "termination-point" here can represent the "port" of a fabric that provides connections to other nodes, such as device internally, another fabric externally and also end hosts.

Details of fabric node and fabric termination point extension will be explained in the following sections.

3.2.2. Fabric node extension

As a network, a fabric itself is composed of set of network elements i.e. devices, and related links. As stated previously, the configuration of a fabric is contained under the "fabric-attribute" container depicted as follows:

```

+--rw fabric-attribute
  +--rw fabric-id?      fabric-id
  +--rw name?           string
  +--rw type?           fabrictype:underlayer-network-type
  +--rw vni-capacity
    | +--rw min?      int32
    | +--rw max?      int32
  +--rw description?    string
  +--rw options
    | +--rw gateway-mode?      enumeration
    | +--rw traffic-behavior?  enumeration
    | +--rw capability-supported*  fabrictype:service-capabilities
  +--rw device-nodes* [device-ref]
    | +--rw device-ref      fabrictype:node-ref
    | +--rw role?           fabrictype:device-role
  +--rw device-links* [link-ref]
    | +--rw link-ref        fabrictype:link-ref
  +--rw device-ports* [port-ref]
    +--rw port-ref          fabrictype:tp-ref
    +--rw port-type?        enumeration
    +--rw bandwidth?        Enumeration

```

As in the module, additional data objects for nodes are introduced by augmenting the "node" list of the network module. New objects include fabric name, type of the fabric, descriptions of the fabric as well as a set of options defined in an "options" container. The options container includes type of the gateway-mode (centralized or distributed) and traffic-behavior (whether acl needed for the traffic).

Also, it defines a list of device-nodes and related links as supporting-nodes to form a fabric network. These device nodes and links are leaf-ref of existing nodes and links in the physical topology. For the device-node, the "role" object is defined to represents the role of the device within the fabric, such as "SPINE" or "LEAF", which should work together with gateway-mode.

3.2.3. Fabric termination-point extension

Since the fabric is considered as a node, in this concept, "termination-points" can represent "ports" of a fabric that connects to other fabrics or end hosts, besides representing ports that connect devices inside the fabric itself.

As such, the "termination-point" in the fabric topology has three roles, including internal TP that connects to devices within a

fabric, external TP that connects to outside network, as well as access TP to end hosts.

A set of "termination-point" indicates all connections of a fabric including its internal connections, interconnections with other fabrics and also connections to end hosts for a DC network.

The structure of fabric ports is as follows:

```
augment /nw:networks/nw:network/nw:node/nt:termination-point:
  +--ro fport-attribute
    +--ro name?          string
    +--ro role?          fabric-port-role
    +--ro type?          fabric-port-type
    +--ro device-port?   tp-ref
    +--ro (tunnel-option)?
      +--:(gre)
        +--ro src-ip?    inet:ip-prefix
        +--ro dest-ip?   inet:ip-address
```

It augments the termination points (in network topology module) with fabric port attributes defined in a container.

New nodes are defined for fabric ports which include name, role of the port within the fabric (internal port, external port to outside network, access port to end hosts), port type (l2 interface, l3 interface etc al). By using the device-port defined as a tp-ref, this fabric port can be mapped to a device node in the underlay network.

Also, a new container for tunnel-options is introduced as well to present the tunnel configuration on the port.

The termination points information are all learnt from the underlay networks but not configured by the fabric topology layer.

4. Fabric YANG Module

```
<CODE BEGINS> file "ietf-fabric-types@2016-09-29.yang"
module ietf-fabric-types {

  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-fabric-types";
  prefix fabriotypes;

  import ietf-inet-types { prefix "inet"; revision-date "2013-07-15"; }
```

```
import ietf-network-topology { prefix nt; }

organization
"IETF I2RS (Interface to the Routing System) Working Group";

contact
"WG Web:      <http://tools.ietf.org/wg/i2rs/ >
WG List:      <mailto:i2rs@ietf.org>

WG Chair:     Susan Hares
               <mailto:shares@ndzh.com>

WG Chair:     Russ White
               <mailto:russ@riw.us>

Editor:       Yan Zhuang
               <mailto:zhuangyan.zhuang@huawei.com>

Editor:       Danian Shi
               <mailto:shidanian@huawei.com>";

description
"This module contains a collection of YANG definitions for Fabric.";

revision "2016-09-29" {
  description
    "Initial revision of faas.";
  reference
    "draft-zhuang-i2rs-yang-dc-fabric-network-topology-02";
}

identity fabric-type {
  description
    "base type for fabric networks";
}

identity vxlan-fabric {
  base fabric-type;
  description
    "vxlan fabric";
}

identity vlan-fabric {
  base fabric-type;
  description
    "vlan fabric";
}
```

```
typedef service-capabilities {
  type enumeration {
    enum ip-mapping {
      description "NAT";
    }
    enum acl-redirect{
      description "acl redirect, which can provide SFC
function";
    }
    enum dynamic-route-exchange{
      description "dynamic route exchange";
    }
  }
  description
    "capability of the device";
}

/*
 * Typedefs
 */
typedef node-ref {
  type instance-identifier;
  description "A reference to a node in topology";
}

typedef tp-ref {
  type instance-identifier;
  description "A reference to a termination point in topology";
}

typedef link-ref {
  type instance-identifier;
  description "A reference to a link in topology";
}

typedef device-role {
  type enumeration {
    enum SPINE {
      description "a spine node";
    }
    enum LEAF {
      description "a leaf node";
    }
    enum BORDER {
      description "a border node";
    }
  }
  default "LEAF";
  description "device role type";
}
```

```
    }

    typedef fabric-port-role {
        type enumeration {
            enum internal {
                description "the port used for devices to access each other.";
            }
            enum external {
                description "the port used for fabric to access outside network.
";
            }
            enum access {
                description "the port used for Endpoint to access fabric.";
            }
            enum reserved {
                description " not decided yet. ";
            }
        }
        description "the role of the physical port ";
    }

    typedef fabric-port-type {
        type enumeration {
            enum layer2interface {
                description "l2 if";
            }
            enum layer3interface {
                description "l3 if";
            }
            enum layer2Tunnel {
                description "l2 tunnel";
            }
            enum layer3Tunnel {
                description "l3 tunnel";
            }
        }
        description
            "fabric port type";
    }

    typedef underlayer-network-type {
        type enumeration {
            enum VXLAN {
                description "vxlan";
            }
            enum TRILL {
                description "trill";
            }
            enum VLAN {
```

```
        description "vlan";
    }
}
    description "";
}

typedef layer2-protocol-type-enum {
    type enumeration {
        enum VLAN{
            description "vlan";
        }
        enum VXLAN{
            description "vxlan";
        }
        enum TRILL{
            description "trill";
        }
        enum NvGRE{
            description "nvgre";
        }
    }
    description "";
}

typedef access-type {
    type enumeration {
        enum exclusive{
            description "exclusive";
        }
        enum vlan{
            description "vlan";
        }
    }
    description "";
}

grouping fabric-port {
    description
        "attributes of a fabric port";
    leaf name {
        type string;
        description "name of the port";
    }
    leaf role {
        type fabric-port-role;
        description "role of the port in a fabric";
    }
    leaf type {
```

```
        type fabric-port-type;
            description "type of the port";
    }
    leaf device-port {
        type tp-ref;
            description "the device port it mapped to";
    }
    choice tunnel-option {
        description "tunnel options";

        case gre {
            leaf src-ip {
                type inet:ip-prefix;
                    description "source address";
            }
            leaf dest-ip {
                type inet:ip-address;
                    description "destination address";
            }
        }
    }
}

grouping route-group {
    description
        "route attributes";
    list route {
        key "destination-prefix";
        description "route list";

        leaf description {
            type string;
            description "Textual description of the route.";
        }
        leaf destination-prefix {
            type inet:ipv4-prefix;
            mandatory true;
            description "IPv4 destination prefix.";
        }
        choice next-hop-options {
            description "choice of next hop options";
            case simple-next-hop {
                leaf next-hop {
                    type inet:ipv4-address;
                    description "IPv4 address of the next hop.";
                }
                leaf outgoing-interface {
                    type nt:tp-id;
                }
            }
        }
    }
}
```



```

        description "Name of the outgoing interface.";
    }
}
}
}
}
grouping port-functions {
    description
        "port functions";

    container port-function {
        description "port functions";
        choice function-type {
            description "type of functions";
            case ip-mapping {
                list ip-mapping-entry {
                    key "external-ip";
                    description "list of NAT entry";
                    leaf external-ip {
                        type inet:ipv4-address;
                        description "external address";
                    }
                    leaf internal-ip {
                        type inet:ipv4-address;
                        description "internal address";
                    }
                }
            }
        }
    }
}
grouping acl-list {
    description "acl list";
    list fabric-acl {
        key fabric-acl-name;
        description "fabric acl list";
        leaf fabric-acl-name {
            type string;
            description "acl name";
        }
    }
}
}
<CODE ENDS>

<CODE BEGINS> file "ietf-fabric-topology@2017-03-10.yang"
module ietf-fabric-topology {

```

```
yang-version 1.1;
namespace "urn:ietf:params:xml:ns:yang:ietf-fabric-topology";
prefix fabric;

import ietf-network { prefix nw; }
import ietf-network-topology { prefix nt; }
import ietf-fabric-types { prefix fabrictype; revision-date "2016-09-29"; }

organization
  "IETF I2RS (Interface to the Routing System) Working Group";

contact
  "WG Web:    <http://tools.ietf.org/wg/i2rs/ >
  WG List:    <mailto:i2rs@ietf.org>

  WG Chair:   Susan Hares
              <mailto:shares@ndzh.com>

  WG Chair:   Russ White
              <mailto:russ@riw.us>

  Editor:     Yan Zhuang
              <mailto:zhuangyan.zhuang@huawei.com>

  Editor:     Danian Shi
              <mailto:shidanian@huawei.com>";

description
  "This module contains a collection of YANG definitions for Fabric.

  Copyright (c) 2016 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).

  This version of this YANG module is part of
  draft-zhuang-i2rs-yang-dc-fabric-network-topology;
  see the RFC itself for full legal notices.";

revision "2017-03-10" {
  description
    "remove the rpcs and add extra attributes";
```

```
        reference
            "draft-zhuang-i2rs-yang-dc-fabric-network-topology-03";
    }
    revision "2016-09-29" {
        description
            "Initial revision of fabric topology.";
        reference
            "draft-zhuang-i2rs-yang-dc-fabric-network-topology-02";
    }

    identity fabric-context {
        description
            "identity of fabric context";
    }

    typedef fabric-id {
        type nw:node-id;
        description
            "An identifier for a fabric in a topology.
            The identifier is generated by compose-fabric RPC.";
    }

    //grouping statements
    grouping fabric-network-type {
        description "Identify the topology type to be fabric.";
        container fabric-network {
            presence "indicates fabric Network";
            description
                "The presence of the container node indicates
                fabric Topology";
        }
    }

    grouping fabric-options {
        description "options for a fabric";

        leaf gateway-mode {
            type enumeration {
                enum centralized {
                    description "centerilized gateway";
                }
                enum distributed {
                    description "distributed gateway";
                }
            }
            default "distributed";
            description "gateway mode";
        }
    }
```

```
    leaf traffic-behavior {
      type enumeration {
        enum normal {
          description "normal";
        }
        enum policy-driven {
          description "policy driven";
        }
      }
      default "normal";
      description "traffic behavior of the fabric";
    }

    leaf-list capability-supported {
      type fabricktype:service-capabilities;
      description
        "supported services of the fabric";
    }
  }

  grouping device-attributes {
    description "device attributes";
    leaf device-ref {
      type fabricktype:node-ref;
      description
        "the device it includes to";
    }
    leaf role {
      type fabricktype:device-role;
      default "LEAF";
      description
        "role of the node";
    }
  }

  grouping link-attributes {
    description "link attributes";
    leaf link-ref {
      type fabricktype:link-ref;
      description
        "the link it includes";
    }
  }

  grouping port-attributes {
    description "port attributes";
    leaf port-ref {
      type fabricktype:tp-ref;
```

```
        description
            "port reference";
    }
    leaf port-type {
        type enumeration {
            enum ETH {
                description "ETH";
            }
            enum SERIAL {
                description "Serial";
            }
        }
        description
            "port type: ethernet or serial";
    }
    leaf bandwidth {
        type enumeration {
            enum 1G {
                description "1G";
            }
            enum 10G {
                description "10G";
            }
            enum 40G {
                description "40G";
            }
            enum 100G {
                description "100G";
            }
            enum 10M {
                description "10M";
            }
            enum 100M {
                description "100M";
            }
            enum 1M {
                description "1M";
            }
        }
        description
            "bandwidth on the port";
    }
}

grouping fabric-attributes {
    description "attributes of a fabric";

    leaf fabric-id {
```

```
        type fabric-id;
            description
                "fabric id";
    }

    leaf name {
        type string;
            description
                "name of the fabric";
    }

    leaf type {
        type fabrictype:underlayer-network-type;
        description
            "The type of physical network that implements th
is fabric.Examples are vlan, and trill.";
    }

        container vni-capacity {
        description "number of vnis the fabric has";
        leaf min {
            type int32;
                description
                    "vni min capacity";
        }

        leaf max {
            type int32;
                description
                    "vni max capacity";
        }
    }

    leaf description {
        type string;
            description
                "description of the fabric";
    }

    container options {
        description "options of the fabric";
        uses fabric-options;
    }

    list device-nodes {
        key device-ref;
        description "include device nodes in the fabric";
        uses device-attributes;
    }
```

```
list device-links {
    key link-ref;
    description "include device links within the fabric";
    uses link-attributes;
}

list device-ports {
    key port-ref;
    description "include device ports within the fabric";
    uses port-attributes;
}

}

// augment statements

augment "/nw:networks/nw:network/nw:network-types" {
description
    "Introduce new network type for Fabric-based logical topology";

    uses fabric-network-type;
}

augment "/nw:networks/nw:network/nw:node" {
    when "/nw:networks/nw:network/nw:network-types/fabric-network" {
description
    "Augmentation parameters apply only for networks
    with fabric topology";
    }
description "Augmentation for fabric nodes created by faas.";

    container fabric-attribute {
        description
            "attributes for a fabric network";

        uses fabric-attributes;
    }
}

augment "/nw:networks/nw:network/nw:node/nt:termination-point" {
    when "/nw:networks/nw:network/nw:network-types/fabric-network" {
description
    "Augmentation parameters apply only for networks
    with fabric topology";
    }
description "Augmentation for port on fabric.";

    container fport-attribute {
```

```
        config false;
            description
                "attributes for fabric ports";
            uses fabrictype:fabric-port;
        }
    }
}
<CODE ENDS>
```

5. Security Consideration

TBD

6. Acknowledgements

7. References

7.1. Normative References

- [I-D.draft-ietf-i2rs-yang-l3-topology]
Clemm, A., Medved, J., Tkacik, T., Liu, X., Bryskin, I.,
Guo, A., Ananthakrishnan, H., Bahadur, N., and V. Beeram,
"A YANG Data Model for Layer 3 Topologies", I-D draft-
ietf-i2rs-yang-l3-topology-04, September 2016.
- [I-D.draft-ietf-i2rs-yang-network-topo]
Clemm, A., Medved, J., Tkacik, T., Varga, R., Bahadur, N.,
and H. Ananthakrishnan, "A YANG Data Model for Network
Topologies", I-D draft-ietf-i2rs-yang-network-topo-06,
September 2016.
- [I-D.draft-ietf-nvo3-vxlan-gpe]
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol
Extension for VXLAN", I-D draft-ietf-i2rs-yang-network-
topo-02, October 2016.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the
Network Configuration Protocol (NETCONF)", RFC 6020,
October 2010.
- [RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991,
July 2013.

[RFC7950] Bjorklund, M., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016.

7.2. Informative References

[I-D.draft-ietf-i2rs-usecase-reqs-summary]
Hares, S. and M. Chen, "Summary of I2RS Use Case Requirements", I-D draft-ietf-i2rs-usecase-reqs-summary-01, May 2015.

Appendix A. Non NMDA -state modules

```
<CODE BEGINS> file "ietf-fabric-topology-state@2017-06-29.yang"
module ietf-fabric-topology-state {

  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-fabric-topology-state";
  prefix sfabric;

  import ietf-network { prefix nw; }
  import ietf-fabric-types { prefix fabrictype; revision-date "2016-09-29"; }
  import ietf-fabric-topology {prefix fp;}
  organization
    "IETF I2RS (Interface to the Routing System) Working Group";

  contact
    "WG Web:      <http://tools.ietf.org/wg/i2rs/ >
    WG List:      <mailto:i2rs@ietf.org>

    WG Chair:     Susan Hares
                  <mailto:shares@ndzh.com>

    WG Chair:     Russ White
                  <mailto:russ@riw.us>

    Editor:        Yan Zhuang
                  <mailto:zhuangyan.zhuang@huawei.com>

    Editor:        Danian Shi
                  <mailto:shidanian@huawei.com>";

  description
    "This module contains a collection of YANG definitions for Fabric topology state for non NMDA.
```

Copyright (c) 2016 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of draft-zhuang-i2rs-yang-dc-fabric-network-topology; see the RFC itself for full legal notices."

```
revision "2017-06-29"{
  description
    "update to NMDA compliant format";
  reference
    "draft-zhuang-i2rs-yang-dc-fabric-network-topology-04";
}

//grouping statements
grouping fabric-network-type {
description "Identify the topology type to be fabric.";
container fabric-network {
  presence "indicates fabric Network";
  description
    "The presence of the container node indicates
    fabric Topology";
}
}

grouping fabric-options {
  description "options for a fabric";

  leaf gateway-mode {
    type enumeration {
      enum centralized {
        description "centerilized gateway";
      }
      enum distributed {
        description "distributed gateway";
      }
    }
    default "distributed";
    description "gateway mode";
  }

  leaf traffic-behavior {
    type enumeration {
      enum normal {
```

```
        description "normal";
    }
    enum policy-driven {
        description "policy driven";
    }
}
default "normal";
        description "traffic behavior of the fabric";
}

leaf-list capability-supported {
    type fabricktype:service-capabilities;
    description
        "supported services of the fabric";
}
}

grouping device-attributes {
    description "device attributes";
    leaf device-ref {
        type fabricktype:node-ref;
        description
            "the device it includes to";
    }
    leaf role {
        type fabricktype:device-role;
        default "LEAF";
        description
            "role of the node";
    }
}

grouping link-attributes {
    description "link attributes";
    leaf link-ref {
        type fabricktype:link-ref;
        description
            "the link it includes";
    }
}

grouping port-attributes {
    description "port attributes";
    leaf port-ref {
        type fabricktype:tp-ref;
        description
            "port reference";
    }
}
```

```
    leaf port-type {
      type enumeration {
        enum ETH {
          description "ETH";
        }
        enum SERIAL {
          description "Serial";
        }
      }
      description
        "port type: ethernet or serial";
    }
    leaf bandwidth {
      type enumeration {
        enum 1G {
          description "1G";
        }
        enum 10G {
          description "10G";
        }
        enum 40G {
          description "40G";
        }
        enum 100G {
          description "100G";
        }
        enum 10M {
          description "10M";
        }
        enum 100M {
          description "100M";
        }
        enum 1M {
          description "1M";
        }
      }
      description
        "bandwidth on the port";
    }
  }
}

grouping fabric-attributes {
  description "attributes of a fabric";

  leaf fabric-id {
    type fp:fabric-id;
    description
      "fabric id";
  }
}
```

```
    }

    leaf name {
        type string;
        description
            "name of the fabric";
    }

    leaf type {
        type fabrictype:underlayer-network-type;
        description
            "The type of physical network that implements th
is fabric.Examples are vlan, and trill.";
    }

    container vni-capacity {
        description "number of vnis the fabric has";
        leaf min {
            type int32;
            description
                "vni min capacity";
        }

        leaf max {
            type int32;
            description
                "vni max capacity";
        }
    }

    leaf description {
        type string;
        description
            "description of the fabric";
    }

    container options {
        description "options of the fabric";
        uses fabric-options;
    }

    list device-nodes {
        key device-ref;
        description "include device nodes in the fabric";
        uses device-attributes;
    }

    list device-links {
        key link-ref;
```

```
        description "include device links within the fabric";
    uses link-attributes;
}

    list device-ports {
    key port-ref;
        description "include device ports within the fabric";
    uses port-attributes;
}
}

// augment statements

augment "/nw:networks/nw:network/nw:network-types" {
description
    "Introduce new network type for Fabric-based logical topology";

    uses fabric-network-type;
}

augment "/nw:networks/nw:network/nw:node" {
    when "/nw:networks/nw:network/nw:network-types/fabric-network" {
    description
        "Augmentation parameters apply only for networks
        with fabric topology.";
    }
    description "Augmentation for fabric nodes.";

    container fabric-attribute-state {
        config false;
        description
            "attributes for a fabric network";

        uses fabric-attributes;
    }
}
}
```

<CODE ENDS>

Authors' Addresses

Yan Zhuang
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: zhuangyan.zhuang@huawei.com

Danian Shi
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: shidanian@huawei.com

Rong Gu
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing, Beijing 100053
China

Email: gurong_cmcc@outlook.com

Hariharan Ananthakrishnan
Packet Design

Email: hari@packetdesign.com