NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2020

M. Boucadair
Orange
Q. Wu
Z. Wang
Huawei
D. King
Lancaster University
C. Xie
China Telecom

November 3, 2019

Framework for Use of ECA (Event Condition Action) in Network Self Management
draft-bwd-netmod-eca-framework-00

Abstract

Event-driven management is meant to provide a useful method to monitor state change of managed objects and resources, and facilitate automatic triggering of a response to events, based on an established set of rules. This would provide rapid autonomic responses to specific conditions, enabling self-management behaviors, including: self-configuration, self-healing, self-optimization, and self-protection.

This document provides a framework that describes the architecture for supporting event-driven management of managed object state across devices. It does not describe specific protocols or protocol extensions needed to realize the objectives and capabilities discussed in the document.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://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 6, 2020.

Copyright Notice

Copyright (c) 2019 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 (https://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. Terminology 4
2. Problem Statement 4
2.1. Defining Network Event and Network Control Logic 4
2.2. Delegating Network Control Logic to Network Device . . . 4
2.3. Executing ECA Script in the Network Device 5
2.4. Event-Driven Notification Handling 6
2.5. Requisite State Information 6
3. Architectural Concepts 7
3.1. What is Defined in ECA Policy? 7
3.2. Where is ECA Script and State Held? 8
3.3. What State is Held? 9
4. Architecture Overview 9
4.1. Telemetry Automation in the Network Device 10
4.2. Detecting and Resolving Policy Conflict 12
4.3. Chain Reaction of Coordinated Events 12
5. Security Considerations 12
6. Acknowledgements 13
7. References 13
7.1. Normative References 13
7.2. Informative References 14
Authors' Addresses 15

1. Introduction

Network management data objects can often take two different values: the value configured by the administrator or an application (configuration) and the value that the device is actually using (operational state). Particularly, these network management data objects can be fetched from various different YANG datastore

[RFC8342] by subscribing to continuous datastore updates [RFC8641] without needing to poll for data periodically.

YANG-Push mechanisms are used to select which data objects are of interest using filters and provide frequent or prompt updates of remote object state, thus allowing (client) applications to maintain a continuous view of operational data and state and enabling a network operator to optimize the system behavior across the whole network to meet objectives and provide some performance guarantees for network services.

Network management may rely upon one or multiple policies to influence management behavior within the system and make sure policies are enforced or executed correctly so that there will no conflict in policies and that the observed behavior is the expected one. Event-driven policy (i.e., ECA Policy [RFC8328]) enables actions being automatically triggered based on when certain events in the network occur while certain conditions hold. YANG Push subscription provides a source for such events.

It is often the case that where Event Condition Action (ECA) is defined is decoupled from where ECA is executed. ECA Engine in the management system or the network device defines one or multiple events corresponding to the workflow management, correlate these events with action triggers and create ECA policy. ECA policy can be enforced either at the management system or pushed to and executed by the network device. Alternative, some of these predefined events can be translated into filter in the YANG push subscription which is in turn used to select data objects that are of interest. When these data objects are streamed out to the destination, both the management system and network device check for the condition when the event is observed. If the condition is satisfied, the ECA script is executed.

Event-driven management (of states of managed objects) across a wide range of devices can be used to monitor state changes of managed objects or resource and automatic trigger of rules in response to events so as to better service assurance for customers and to provide rapid autonomic response that can exhibit self-management properties including self-configuration, self-healing, self-optimization, and self-protection.

This document provides a framework that describes the architecture for supporting such event-driven management.

This document does not describe specific protocols or protocol extensions needed to realize the objectives and capabilities discussed in the document.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Problem Statement

2.1. Defining Network Event and Network Control Logic

Datastores are used by network management protocols such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Operational state data objects, in the operational state datastore, provide network visibility to the actual state of the network, and ensure the network is running efficiently.

The network event are used to keep track of state of changes associated with one of multiple operational state data objects in the network device. Typical examples of network event include a fault, an alarm, a change in network state, network security threat, hardware malfunction, buffer utilization crossing a threshold, network connection setup, and an external input to the system.

To control which state a network device should be in or is allowed to be in at any given time, a set of conditions and actions are defined and correlated with network events, which constitute an event-driven policy or network control logic.

YANG Push subscription allows client applications to select which datastore nodes are of interest and provides source of network events. The NETCONF client can define event-based policy based on YANG Push subscription data source or some other data source.

2.2. Delegating Network Control Logic to Network Device

Usually the NETCONF clients subscribe to continuous datastore updates and rely on event notifications sent to the NETCONF client to check for the condition so that reaction to many network events may be very slow in the face of communication glitches between the client and the sever. Such solution doesn’t scale well.

It is more desirable in many circumstances to delegate all event response behaviors (e.g., recover from network failure, instruct the network to control congestion) to the NETCONF server so that the network can react to network change as quickly as the event is detected.

The event response behaviors delegation can be done using YANG push subscription filter enhancements, e.g., define a new filter to allow the NETCONF client send updates only when the value falls within a certain range. Another example is to define a filter to allow the NETCONF client send updates only when the value exceeds a certain threshold for the first time but not again until the threshold is cleared. In the latter case, additional state is required.

In addition, the event response behavior delegation can be done by pushing ECA policy to the network device. Similar to YANG Push subscription filter, the ECA approach also includes filter and defines it as Event and Condition separately in the ECA policy model. Different from using YANG Push subscription filter, ECA allow a group of events to be observed, allow multiple actions to be triggered, e.g., sending log report notification, add or remove multiple YANG Push subscriptions.

2.3. Executing ECA Script in the Network Device

When the YANG Push subscription filter or ECA policy is pushed to the server, the server is expected to register the event conveyed in the YANG push subscription filter or Event-driven policy, generate server specific script. With a server specific script, the server can manipulate various network resources autonomously.

After the event registration, the server subscribes to its own publications encapsulated in the event notification with respect to all events that are associated with ECA Policy so that the publication is intercepted and all events specified in the ECA policy model are continuously monitored by the server before the publication is encapsluated in the event notification and sent to the YANG Push subscription's client. At the moment of event detection, the server loads the operational state data object filtered by the YANG Push subscription's filters or ECA policy into the auto-configured ECA's event and execute the ECA's associated condition-action chain.

The condition is associated with an ECA event and evaluated only within event threads triggered by the event detection, and the action corresponds to a set of statements that may trigger state changes in the device or publication content changes in the Event subscription and could be various different operations to be carried out by the server:

- Configuration data object reconfiguration;
- ECA Log report Notification;
- Add or remove one or multiple YANG Push Subscriptions;

- Invoke another Event in the same network device or different network device.

2.4. Event-Driven Notification Handling

ECA notifications are the only ECA actions that directly interact and hence need to be unambiguously understood by the client.

ECA notification can be sent when the client may find any interesting about the associated event with all the logic to compute said data (e.g., datastore content changes history, median values), and delegate computation task to the server via an ECA script.

When a "Send ECA notification" action is configured as an ECA Action, the client may receive different ECA notification associated with the same event or different events, YANG Push Publication will also be sent through Event notification. Therefore it is important for client to correlate of events and ECA notifications received from the server.

When ECA notification and YANG Push Publication are both pushed to the client, the client can execute client specific script generated in the same way as the server does and manipulate various network resources autonomously remotely. However the network resource can not be manipulated twice in both client and the server. Therefore policy conflict should be avoided or resolved.

2.5. Requisite State Information

A ECA policy rule is read as: When event occurs in a situation where condition is true, then action is executed. The ECA associated state is used to indicate when Events are triggered and what actions must be performed on the occurrence of an event.

A simple information model for one piece of the ECA associated state is as follows:

```
{
  event name;
  start time;
  end time;
  threshold value;
  event occurrence times
}
```

The event that is observed could be a fault, an alarm, a change in network state, network security threats, hardware malfunctions,

buffer utilization exceeding a threshold. For any of the aforementioned events, multiple actions may be triggered.

3. Architectural Concepts

3.1. What is Defined in ECA Policy?

ECA Event is a change of datastore operational state. Each policy rule consists of a set of conditions and a set of actions. Policy rules may be aggregated into policy groups. These groups may be nested, to represent a hierarchy of policies.

ECA Condition is evaluated to TRUE or FALSE logical expression. ECA condition is specified as a hierarchy of comparisons and logical combinations of thereof, which allows for configuring logical hierarchies. One of ECA condition example is logical hierarchies specified in a form of:

<target><relation><arg>
where target represent managed data object while arg represent either constant/enum/identity, Policy variable or pointed by XPath data store node or sub-tree,

relation is one of the comparison operations from the set: ==, !=, >, <, >=, <=

Logical calculation between multiple trigger conditions:

The YANG language cannot clearly describe complex logical operations between different condition lists under the same event, for example, (condition A & condition B) or condition C.

By default, the ECA model performs logic "AND" operation between different conditions in the same Event. That is, event is triggered when different conditions are met at the same time. For example,

Event A consists of two conditions:
o Condition A;
o Condition B;
If Condition A AND Condition B is met;
Event A is triggered;
Action A is executed.

For the logic "OR" operation between different conditions, the conditions can be defined in different events. If the corresponding event is triggered, the same action is executed. For example,

Event A is triggered on Condition A.
Event B is triggered on Condition B.
If Condition A is met;
Event A is triggered;
Action A is executed.
If Condition B is met;
Event B is triggered;
Action A is executed.

ECA Action is one of the following operations to be carried out by a server:

- Configuration data object reconfiguration
- ECA Log report Notification
- Add or remove one or multiple YANG Push Subscriptions
- Invoke another Event in the same network device or different network device

In case of one event triggering another event, a set of events can be grouped together and executed, in a coordinated manner. The action associated with the event can be executed in the same network device or in different network devices. In the latter case, events executed by different network devices need to coordinate as a group to fulfil a task, previously set.

3.2. Where is ECA Script and State Held?

The ECA state information described in Section 2.5 and associated ECA script has to be held somewhere. There are two locations where this applies:

- in a central controller where decisions about resource adjustment are made;
- in the network nodes where the resources exist.

The first of these locations have a good visibility of the whole network or information of the flow packets are going to take through the entire network, but requires a centralized, searchable repository of all network information that can be used for diagnostics, service assurance, maintenance or audit purposes. The response to network event can be slow since all monitored data objects from large amount of network devices need to be sent and correlated at the point where decisions about resource adjustment are made, less alone multiple network event triggering a single action.

Conversely, if the ECA state and associated ECA script is held in the network nodes, it makes policing of resource adjustment easier. It means many points in the network can have immediate response to network event. The limitation is the configurations and state of a particular device does not have the visibility of the whole network or information of the flow packets are going to take through the entire network, so they provide little insight into network level policy-related behavior.

3.3. What State is Held?

As already described, the network control logic associated with ECA script needs access to ECA state table. It stores network events pushed from YANG push subscription or ECA policy model, threshold value it set for observed network management data object.

In addition, when the event needs to be continuously monitored, the Event scheduling information such as start time, end time can be included.

In case of sending updates only when the value exceeds a certain threshold for the first time but not again until the threshold is cleared, a threshold clear flag is also needed.

In case of monitoring the data change or data change rate, for example, YANG Push On-Change mode [RFC8641] or ECA Threshold Test [I.D-wwx-netmod-event-yang], the ECA state table need to store history state to check the condition to be satisfied and determine the current state.

4. Architecture Overview

The architectural considerations and conclusions described in the previous section lead to the architecture described in this section and illustrated in Figure 1. The interfaces and interactions shown in the figure and labeled (a) through (f) are described in Section 4.1.

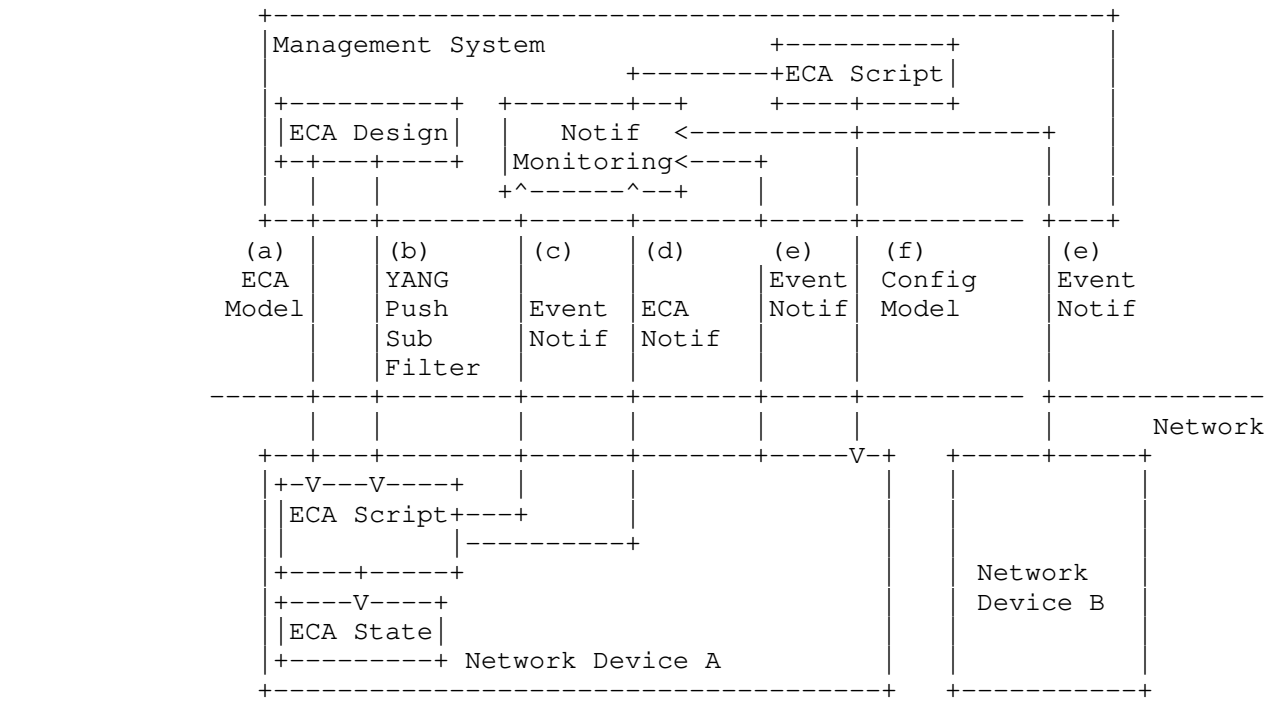


Figure 1.Reference Architecture for Use of ECA and Network Self Management

4.1. Telemetry Automation in the Network Device

As shown in Figure 1, some component in the management system defines and designs ECA Policy rule. This may be invoked by a Service assurance application or device fault self-management application. We show this component on the figure as the "ECA Design", and it extracts Event and Condition in the ECA model and fill into YANG Push Subscription as filter. When YANG Push subscription filter is pushed down to the network device, ECA script can be generated automatically from it (ECA script can also be generated in the management system and downloaded to the network device). The YANG Push subscription request, indicated on Figure 1 by the arrow (b), includes all of the parameters of network management data objects that the requester wishes to be supplied, such as filter node, threshold value, start time, and end time. Note that the requester in this case may be the management system shown in the figure or a distinct system such as data collector.

The network device registers network event that is corresponding to the filter carried in the YANG Push Subscription and enters the

network event in its ECA state and then the server subscribes to its own continuous datastore updates in the operational state datastore that is encapsulated in the event notification as publication to the YANG Push subscriber.

Upon the network event is detected, the server intercepts the publication of subscribed data and loads the operational state data object in the operational state datastore into the auto-configured ECA's event and execute the ECA's associated condition. When ECA Condition is evaluated to TRUE, the operational state data objects will be filtered and the remaining data objects will be entered back into the publication of subscribed data and encapsulated in the event notification (c) and sent to notification monitoring component in the management system.

The notification monitoring component may further derive some new ECA policy rule and fed into ECA Design component. The remaining procedure is same as the procedure starting from (b).

Alternatively, the ECA design component can push ECA model directly with additional actions included (a) to the network device, ECA script is generated automatically from ECA model. The ECA model request, indicated on Figure 1 by the arrow (a), includes additional action parameters besides one included in the YANG Push subscription request.

The network device register network event that is corresponding to the ECA carried in the ECA request and enter them in its ECA state and then the server subscribes to its own continuous datastore updates in the operational state datastore that is encapsulated in the event notification as publication to the YANG Push subscriber.

Upon the network event is detected, the server loads the operational state data object in the operational state datastore into the auto-configured ECA's event and execute the ECA's associated condition. Different from YANG Push subscription filter, the server will not intercept the publication of subscribed data. Instead, it allows the server to trigger a set of actions associated with the network event, e.g., send ECA log report notification, add/remove YANG push subscription, reconfigure the network management data object within the control of the server. After all actions are executed, one or multiple separate ECA notifications (d) can be sent to the notification monitoring component in the management system, the remaining procedure is same as YANG Push subscription case.

Conversely,when, network level policy-related behavior became necessary, once a subscription has been set up, event notification message associated with the subscription from different network

device will be sent to the notification monitoring component in the management system(e), which in turn trigger network behavior change on the network device via configuration model (f).

4.2. Detecting and Resolving Policy Conflict

There are two possible places where policy conflict can take places:

1. An event triggers multiple actions in the network device that cannot occur together as specified by the system administrator.

2. Multiple ECA notifications or multiple combination of ECA notification and Event notification lead to generate ECA policy that cannot occur together.

In both case, policy conflict can be addressed by policy conflict detection mechanism and Policy validation mechanism.

4.3. Chain Reaction of Coordinated Events

In some cases events executed by the same or different network devices can be executed in a coordinated manner. To make sure these network devices coordinate on some task or a group of events coordinate in the same network device, these events on the same or different network devices need to be pre-configured to work together. During capability negotiation phase, the management system should know what each network device supports, which event may take action, and what condition on which event. So ECA model with multiple events can be configured on the network device to allow one event be triggered by another event configured on the same network device.

5. Security Considerations

The framework described in this document for supporting autonomic event-driven self-management will require consideration of potential security and operational requirements, and ensure best security practices and methods are applied.

Key security considerations that will be discussed in future versions of this document, include:

- Authentication of ECA programming requests;

- Application of suitable authorization methods when enabling ECA functions;

- Securing ECA communication channels;

- Locking ECA device config and state databases;
- Mitigation, and negation, of ECA functional component attacks;
- Logging and auditing of ECA transactions;
- Maintaining ECA device confidentially.

6. Acknowledgements

This work has benefited from the discussions of ECA Policy over the years. In particular, the SUPA project [https://datatracker.ietf.org/wg/supa/about/] provided approaches to express high-level, possibly network-wide policies to a network management function (within a controller, an orchestrator, or a network element).

Igor Bryskin, Xufeng Liu, Alexander Clemm, Henk Birkholz, Tianran Zhou contributed to an earlier version of [GNCA]. We would like to thank the authors of that document on event response behaviors delegation for material that assisted in thinking that helped develop this document.

Finally, the authors would like to thank David Hutchison and Mehdi Bezahaf at Lancaster University, Phil Eardley and Andy Reid at British Telecom, for their input and applicability of ECA device self management to the Next Generation Converged Digital Infrastructure (NG-CDI) project.

7. References

7.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

7.2. Informative References

[I-D.bryskin-netconf-automation-yang] Bryskin, I., Liu, X., Clemm, A., Birkholz, H., and T. Zhou, "Generalized Network Control Automation YANG Model", draft-bryskin-netconf-automation-yang-03 (work in progress), July 2019.

[I-D.clemm-netmod-push-smart-filters] Clemm, A., Voit, E., Liu, X., Bryskin, I., Zhou, T., Zheng, G., and H. Birkholz, "Smart Filters for Push Updates", draft-clemm-netmod-push-smart-filters-01 (work in progress), October 2018.

[I-D.clemm-nmrg-dist-intent] Clemm, A., Ciavaglia, L., Granville, L., and J. Tantsura, "Intent-Based Networking - Concepts and Overview", draft-clemm-nmrg-dist-intent-02 (work in progress), July 2019.

[I-D.wwx-netmod-event-yang] Wang, Z., WU, Q., Xie, C., Bryskin, I., Liu, X., Clemm, A., Birkholz, H., and T. Zhou, "A YANG Data model for ECA Policy Management", draft-wwx-netmod-event-yang-04 (work in progress), November 2019.

[RFC8328] Liu, W., Xie, C., Strassner, J., Karagiannis, G., Klyus, M., Bi, J., Cheng, Y., and D. Zhang, "Policy-Based Management Framework for the Simplified Use of Policy Abstractions (SUPA)", RFC 8328, DOI 10.17487/RFC8328, March 2018, <https://www.rfc-editor.org/info/rfc8328>.

[RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero Touch Provisioning (SZTP)", RFC 8572, DOI 10.17487/RFC8572, April 2019, <https://www.rfc-editor.org/info/rfc8572>.

Authors' Addresses

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

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

Email: bill.wu@huawei.com

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

Email: wangzitao@huawei.com

Daniel King
Lancaster University
Bailrigg, Lancaster LA1 4YW
UK

Email: d.king@lancaster.ac.uk

Chongfeng Xie
China Telecom
Beijing
China

Email: xiechf.bri@chinatelecom.cn

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2020

C. Hopps
LabN Consulting, L.L.C.

November 4, 2019

YANG Geo Location
draft-ietf-netmod-geo-location-02

Abstract

This document defines a generic geographical location object YANG grouping. The geographical location grouping is intended to be used in YANG models for specifying a location on or in reference to the Earth or any other astronomical object.

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 https://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 7, 2020.

Copyright Notice

Copyright (c) 2019 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 (https://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. Terminology 3
2. The Geo Location Object 3
2.1. Frame of Reference 3
2.2. Location 4
2.3. Motion 4
2.4. Nested Locations 5
2.5. Non-location Attributes 5
2.6. Tree 5
3. YANG Module 6
4. ISO 6709:2008 Conformance 11
5. Usability 12
5.1. Portability 13
5.1.1. IETF URI Value 13
5.1.2. W3C 13
5.1.3. Geography Markup Language (GML) 15
5.1.4. KML 15
6. IANA Considerations 16
6.1. Geodetic System Value Registry 16
7. Security Considerations 17
8. References 18
8.1. Normative References 18
8.2. Informative References 19
Appendix A. Examples 19
Appendix B. Acknowledgements 22
Author's Address 23

1. Introduction

In many applications we would like to specify the location of something geographically. Some examples of locations in networking might be the location of data center, a rack in an internet exchange point, a router, a firewall, a port on some device, or it could be the endpoints of a fiber, or perhaps the failure point along a fiber.

Additionally, while this location is typically relative to The Earth, it does not need to be. Indeed it is easy to imagine a network or device located on The Moon, on Mars, on Enceladus (the moon of Saturn) or even a comet (e.g., 67p/churyumov-gerasimenko).

Finally, one can imagine defining locations using different frames of reference or even alternate systems (e.g., simulations or virtual realities).

This document defines a "geo-location" YANG grouping that allows for all of the above data to be captured.

This specification conforms to [ISO.6709.2008].

The YANG data model described in this document conforms to the Network Management Datastore Architecture defined in [RFC8342].

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. The Geo Location Object

2.1. Frame of Reference

The frame of reference ("reference-frame") defines what the location values refer to and their meaning. The referred to object can be any astronomical body. It could be a planet such as The Earth or Mars, a moon such as Enceladus, an asteroid such as Ceres, or even a comet such as 1P/Halley. This value is specified in "astronomical-body" and is defined by the International Astronomical Union (<http://www.iau.org>), The default "astronomical-body" value is "earth".

In addition to identifying the astronomical body we also need to define the meaning of the coordinates (e.g., latitude and longitude) and the definition of 0-height. This is done with a "geodetic-datum" value. The default value for "geodetic-datum" is "wgs-84" (i.e., the World Geodetic System, [WGS84]), which is used by the Global Positioning System (GPS) among many others. We define an IANA registry for specifying standard values for the "geodetic-datum".

In addition to the "geodetic-datum" value we allow refining the coordinate and height accuracy using "coord-accuracy" and "height-accuracy" respectively. When specified these values override the defaults implied by the "geodetic-datum" value.

Finally, we define an optional feature which allows for changing the system for which the above values are defined. This optional feature adds an "alternate-system" value to the reference frame. This value is normally not present which implies the natural universe is the system. The use of this value is intended to allow for creating virtual realities or perhaps alternate coordinate systems. The definition of alternate systems is outside the scope of this document.

2.2. Location

This is the location on or relative to the astronomical object. It is specified using 2 or 3 coordinates values. These values are given either as "latitude", "longitude", and an optional "height", or as Cartesian coordinates of "x", "y" and an optional "z". For the standard location choice "latitude" and "longitude" are specified as fractions of decimal degrees, and the "height" value is in fractions of meters. For the Cartesian choice "x", "y" and "z" are in fractions of meters. In both choices the exact meanings of all of the values are defined by the "geodetic-datum" value in the Section 2.1.

2.3. Motion

Support is added for objects in relatively stable motion. For objects in relatively stable motion the grouping provides a 3-dimensional vector value. The components of the vector are "v-north", "v-east" and "v-up" which are all given in fractional meters per second. The values "v-north" and "v-east" are relative to true-north as defined by the reference frame for the astronomical body, "v-up" is perpendicular to the plane defined by "v-north" and "v-east", and is pointed away from the center of mass.

To derive the 2-dimensional heading and speed one would use the following formulas:

$$speed = \sqrt{v_{north}^{2} + v_{east}^{2}}$$

$$heading = \arctan(v_{east} / v_{north})$$

For some applications that demand high accuracy, and where the data is infrequently updated this velocity vector can track very slow movement such as continental drift.

Tracking more complex forms of motion is outside the scope of this work. The intent of the grouping being defined here is to identify where something is located, and generally this is expected to be somewhere on or relative to the Earth (or another astronomical body). At least two options are available to YANG models that wish to use this grouping with objects that are changing location frequently in non-simple ways, they can add additional motion data to their model directly, or if the application allows it can require more frequent queries to keep the location data current.

2.4. Nested Locations

When locations are nested (e.g., a building may have a location which houses routers that also have locations) the module using this grouping is free to indicate in its definition that the "reference-frame" is inherited from the containing object so that the "reference-frame" need not be repeated in every instance of location data.

2.5. Non-location Attributes

During the development of this module, the question of whether it would support data such as orientation arose. These types of attributes are outside the scope of this grouping because they do not deal with a location but rather describe something more about the object that is at the location. Module authors are free to add these non-location attributes along with their use of this location grouping.

2.6. Tree

The following is the YANG tree diagram [RFC8340] for the geo-location grouping.

```
module: ietf-geo-location
  grouping geo-location
    +-- geo-location
       +-- reference-frame
       |  +-- alternate-system?    string {alternate-systems}?
       |  +-- astronomical-body?   string
       |  +-- geodetic-system
       |     +-- geodetic-datum?    string
       |     +-- coord-accuracy?    decimal64
       |     +-- height-accuracy?   decimal64
       +-- (location)?
       |  +--:(ellipsoid)
       |  |  +-- latitude?    degrees
       |  |  +-- longitude?   degrees
       |  |  +-- height?      decimal64
       |  +--:(cartesian)
       |     +-- x?           decimal64
       |     +-- y?           decimal64
       |     +-- z?           decimal64
       +-- velocity
       |  +-- v-north?   decimal64
       |  +-- v-east?    decimal64
       |  +-- v-up?      decimal64
       +-- timestamp?         types:date-and-time
```

Figure 1: Geo Location YANG tree diagram.

3. YANG Module

```
<CODE BEGINS> file "ietf-geo-location@2019-02-17.yang"
module ietf-geo-location {
  namespace "urn:ietf:params:xml:ns:yang:ietf-geo-location";
  prefix geo;
  import ietf-yang-types { prefix types; }

  organization
    "IETF NETMOD Working Group (NETMOD)";
  contact
    "Christian Hopps <chopps@chopps.org>";

  // RFC Ed.: replace XXXX with actual RFC number and
  // remove this note.

  description
    "This module defines a grouping of a container object for
     specifying a location on or around an astronomical object (e.g.,
     The Earth).
```

```
        Copyright (c) 2019 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
        (https://trustee.ietf.org/license-info).

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

        // RFC Ed.: replace XXXX with actual RFC number and
        // remove this note.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
        NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
        'MAY', and 'OPTIONAL' in this document are to be interpreted as
        described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
        they appear in all capitals, as shown here.";

    revision 2019-02-17 {
      description "Initial Revision";
      reference "RFC XXXX: YANG Geo Location";
    }

    typedef degrees {
      type decimal64 {
        fraction-digits 16;
      }
      units "decimal degrees";
      description "Coordinate value.";
    }

    feature alternate-systems {
      description
        "This feature means the device supports specifying locations
         using alternate systems for reference frames.";
    }

    grouping geo-location {
     description
        "Grouping to identify a location on an astronomical object.";

      container geo-location {
        description
```

```
          "A location on an astronomical body (e.g., The Earth)
           somewhere in a universe.";

        container reference-frame {
          description
            "The Frame of Reference for the location values.";

          leaf alternate-system {
            if-feature alternate-systems;
            type string;
            description
              "The system in which the astronomical body and
               geodetic-datum is defined. Normally, this value is not
               present and the system is the natural universe; however,
               when present this value allows for specifying alternate
               systems (e.g., virtual realities). An alternate-system
               modifies the definition (but not the type) of the other
               values in the reference frame.";
          }
          leaf astronomical-body {
            type string {
              pattern '[ -@\[-\^_-~]*';
            }
            default "earth";
            description
              "An astronomical body as named by the International
               Astronomical Union (IAU) or according to the alternate
               system if specified. Examples include 'sun' (our star),
               'earth' (our planet), 'moon' (our moon), 'enceladus' (a
               moon of Saturn), 'ceres' (an asteroid),
               '67p/churyumov-gerasimenko (a comet). The value should
               be comprised of all lower case ASCII characters not
               including control characters (i.e., values 32..64, and
               91..126). Any preceding 'the' in the name should not
               be included.";
          }
          container geodetic-system {
            description
              "The geodetic system of the location data.";
            leaf geodetic-datum {
              type string {
                pattern '[ -@\[-\^_-~]*';
              }
              default "wgs-84";
              description
                "A geodetic-datum defining the meaning of latitude,
                 longitude and height. The default is 'wgs-84' which is
                 used by the Global Positioning System (GPS). The value
```

```
                SHOULD be comprised of all lower case ASCII characters
                not including control characters (i.e., values 32..64,
                and 91..126). The IANA registry further restricts the
                value by converting all spaces (' ') to dashes ('-')";
            }
            leaf coord-accuracy {
              type decimal64 {
                fraction-digits 6;
              }
              description
                "The accuracy of the latitude longitude pair. When
                 coord-accuracy is specified it overrides the
                 geodetic-datum implied accuracy. If Cartesian
                 coordinates are in use this accuracy corresponds to
                 the X and Y components";
            }
            leaf height-accuracy {
              type decimal64 {
                fraction-digits 6;
              }
              units "meters";
              description
                "The accuracy of height value. When specified it
                 overrides the geodetic-datum implied default. If
                 Cartesian coordinates ar in use this accuracy
                 corresponds to the Z component.";
            }
            // May wish to allow for height to be relative.
            // If so need to decide if we have a boolean (to ground)
            // or an enumeration (e.g., local ground, sea-floor,
            // ground floor, containing object, ...) or even allow
            // for a string for most generic but least portable
            // comparable
            // leaf height-relative {
            // }
          }
        }
        choice location {
          description
            "The location data either in lat/long or Cartesian values";
          case ellipsoid {
            leaf latitude {
              type degrees;
              description
                "The latitude value on the astronomical body. The
                 definition and precision of this measurement is
                 indicated by the reference-frame value.";
            }
```

```
            leaf longitude {
              type degrees;
              description
                "The longitude value on the astronomical body. The
                 definition and precision of this measurement is
                 indicated by the reference-frame.";
            }
            leaf height {
              type decimal64 {
                fraction-digits 6;
              }
              units "meters";
              description
                "Height from a reference 0 value. The precision and '0'
                 value is defined by the reference-frame.";
            }
          }
          case cartesian {
            leaf x {
              type decimal64 {
                fraction-digits 6;
              }
              description
                "The X value as defined by the reference-frame.";
            }
            leaf y {
              type decimal64 {
                fraction-digits 6;
              }
              description
                "The Y value as defined by the reference-frame.";
            }
            leaf z {
              type decimal64 {
                fraction-digits 6;
              }
              units "meters";
              description
                "The Z value as defined by the reference-frame.";
            }
          }
        }
        container velocity {
          description
            "If the object is in motion the velocity vector describes
             this motion at the the time given by the timestamp.";

          leaf v-north {
```

```
            type decimal64 {
              fraction-digits 12;
            }
            units "meters per second";
            description
              "v-north is the rate of change (i.e., speed) towards
               truth north as defined by the ~geodetic-system~.";
          }

          leaf v-east {
            type decimal64 {
              fraction-digits 12;
            }
            units "meters per second";
            description
              "v-east is the rate of change (i.e., speed) perpendicular
               to truth-north as defined by the ~geodetic-system~.";
          }

          leaf v-up {
            type decimal64 {
              fraction-digits 12;
            }
            units "meters per second";
            description
              "v-up is the rate of change (i.e., speed) away from the
               center of mass.";
          }
        }
        leaf timestamp {
          type types:date-and-time;
          description "Reference time when location was recorded.";
        }
      }
    }
  }
  <CODE ENDS>
```

4. ISO 6709:2008 Conformance

[ISO.6709.2008] provides an appendix with a set of tests for conformance to the standard. The tests and results are given in the following table along with an explanation of non-applicable tests.

Test	Description	Pass Explanation
A.1.2.1	elements reqd. for a geo. point location	CRS is always indicated
A.1.2.2	Description of a CRS from a register	CRS register is defined
A.1.2.3	definition of CRS	N/A - Don't define CRS
A.1.2.4	representation of horizontal position	lat/long values conform
A.1.2.5	representation of vertical position	height value conforms
A.1.2.6	text string representation	N/A - No string format

Conformance Test Results

For test "A.1.2.1" the YANG geo location object either includes a CRS ("reference-frame") or has a default defined ([WGS84]).

For "A.1.2.3" we do not define our own CRS, and doing so is not required for conformance.

For "A.1.2.6" we do not define a text string representation, which is also not required for conformance.

5. Usability

The geo-location object defined in this document and YANG module have been designed to be usable in a very broad set of applications. This includes the ability to locate things on astronomical bodies other than The Earth, and to utilize entirely different coordinate systems and realities.

Many systems make use of geo-location data, and so it's important to be able describe this data using this geo-location object defined in this document.

5.1. Portability

In order to verify portability while developing this module the following standards and standard APIs and were considered.

5.1.1. IETF URI Value

[RFC5870] defines a standard URI value for geographic location data. It includes the ability to specify the "geodetic-value" (it calls this "crs") with the default being "wgs-84" [WGS84]. For the location data it allows 2 to 3 coordinates defined by the "crs" value. For accuracy it has a single "u" parameter for specifying uncertainty. The "u" value is in fractions of meters and applies to all the location values. As the URI is a string, all values are specifies as strings and so are capable of as much precision as required.

URI values can be mapped to and from the YANG grouping, with the caveat that some loss of precision (in the extremes) may occur due to the YANG grouping using decimal64 values rather than strings.

5.1.2. W3C

See <https://w3c.github.io/geolocation-api/#dom-geolocationposition>.

W3C Defines a geo-location API in [W3CGEO]. We show a snippet of code below which defines the geo-location data for this API. This is used by many application (e.g., Google Maps API).

```
interface GeolocationPosition {
  readonly attribute GeolocationCoordinates coords;
  readonly attribute DOMTimeStamp timestamp;
};

interface GeolocationCoordinates {
  readonly attribute double latitude;
  readonly attribute double longitude;
  readonly attribute double? altitude;
  readonly attribute double accuracy;
  readonly attribute double? altitudeAccuracy;

  readonly attribute double? speed;
};
```

Figure 2: Snippet Showing Geo-Location Definition

5.1.2.1. Compare with YANG Model

Field	Type	YANG	Type
accuracy	double	coord-accuracy	dec64 fr 6
altitude	double	height	dec64 fr 6
altitudeAccuracy	double	height-accuracy	dec64 fr 6
heading	double	heading	dec64 fr 16
latitude	double	latitude	dec64 fr 16
longitude	double	longitude	dec64 fr 16
speed	double	speed	dec64 fr 12
timestamp	DOMTimeStamp	timestamp	string

accuracy (double):
Accuracy of "latitude" and "longitude" values in meters.

altitude (double):
Optional height in meters above the [WGS84] ellipsoid.

altitudeAccuracy (double):
Optional accuracy of "altitude" value in meters.

heading (double):
Optional Direction in decimal deg from true north increasing clock-wise.

latitude, longitude (double):
Standard lat/long values in decimal degrees.

speed (double):
Speed along heading in meters per second.

timestamp (DOMTimeStamp):
Specifies milliseconds since the Unix EPOCH in 64 bit unsigned integer. The YANG model defines the timestamp with arbitrarily large precision by using a string which encompasses all representable values of this timestamp value.

W3C API values can be mapped to the YANG grouping, with the caveat that some loss of precision (in the extremes) may occur due to the YANG grouping using decimal64 values rather than doubles.

Conversely, only YANG values for The Earth using the default "wgs-84" [WGS84] as the "geodetic-datum", can be directly mapped to the W3C values, as W3C does not provide the extra features necessary to map the broader set of values supported by the YANG grouping.

5.1.3. Geography Markup Language (GML)

ISO adopted the Geography Markup Language (GML) defined by OGC 07-036 as [ISO.19136.2007]. GML defines, among many other things, a position type "gml:pos" which is a sequence of "double" values. This sequence of values represent coordinates in a given CRS. The CRS is either inherited from containing elements or directly specified as attributes "srsName" and optionally "srsDimension" on the "gml:pos".

GML defines an Abstract CRS type which Concrete CRS types derive from. This allows for many types of CRS definitions. We are concerned with the Geodetic CRS type which can have either ellipsoidal or Cartesian coordinates. We believe that other non-Earth based CRS as well as virtual CRS should also be representable by the GML CRS types as well.

Thus GML "gml:pos" values can be mapped directly to the YANG grouping, with the caveat that some loss of precision (in the extremes) may occur due to the YANG grouping using decimal64 values rather than doubles.

Conversely, YANG grouping values can be mapped to GML as directly as the GML CRS available definitions allow with a minimum of Earth-based geodetic systems fully supported.

GML also defines an observation value in "gml:Observation" which includes a timestamp value "gml:validTime" in addition to other components such as "gml:using" "gml:target" and "gml:resultOf". Only the timestamp is mappable to and from the YANG grouping. Furthermore "gml:validTime" can either be an Instantaneous measure ("gml:TimeInstant") or a time period ("gml:TimePeriod"). Only the instantaneous "gml:TimeInstant" is mappable to and from the YANG grouping.

5.1.4. KML

KML 2.2 [KML22] (formerly Keyhole Markup Language) was submitted by Google to Open Geospatial Consortium (OGC) <https://www.opengeospatial.org/> and was adopted. The latest

version as of this writing is KML 2.3 [KML23]. This schema includes geographic location data in some of it's objects (e.g., <kml:Point or <kml:Camera> objects). This data is provided in string format and corresponds to the [W3CGEO] values. The timestamp value is also specified as a string as in our YANG grouping.

KML has some special handling for the height value useful for visualization software, "kml:altitudeMode". These values for "kml:altitudeMode" include indicating the height is ignored ("clampToGround"), in relation to the locations ground level ("relativeToGround"), or in relation to the geodetic datum ("absolute"). The YANG grouping can directly map the ignored and absolute cases, but not the relative to ground case.

In addition to the "kml:altitudeMode" KML also defines two seafloor height values using "kml:seaFloorAltitudeMode". One value is to ignore the height value ("clampToSeaFloor") and the other is relative ("relativeToSeaFloor"). As with the "kml:altitudeMode" value, the YANG grouping supports the ignore case but not the relative case.

The KML location values use a geodetic datum defined in Annex A by the GML Coordinate Reference System (CRS) [ISO.19136.2007] with identifier "LonLat84_5773". The altitude value for KML absolute height mode is measured from the vertical datum specified by [WGS84].

Thus the YANG grouping and KML values can be directly mapped in both directions (when using a supported altitude mode) with the caveat that some loss of precision (in the extremes) may occur due to the YANG grouping using decimal64 values rather than strings. For the relative height cases the application doing the transformation is expected to have the data available to transform the relative height into an absolute height which can then be expressed using the YANG grouping.

6. IANA Considerations

6.1. Geodetic System Value Registry

This registry allocates names for standard geodetic systems. Often these values are referred to using multiple names (e.g., full names or multiple acronyms values). The intent of this registry is to provide a single standard value for any given geodetic system.

The values SHOULD use an acronym when available, they MUST be converted to lower case, and spaces MUST be changed to dashes "-".

Each entry should be sufficient to define the 3 coordinate values (2 if height is not required). So for example the "wgs-84" is defined

as WGS-84 with the geoid updated by at least [EGM96] for height values. Specific entries for [EGM96] and [EGM08] are present if a more precise definition of the data is required.

It should be noted that [RFC5870] also creates a registry for Geodetic Systems (it calls CRS); however, this registry has a very strict modification policy. The authors of [RFC5870] have the stated goal of making CRS registration hard to avoid proliferation of CRS values. As our module defines alternate systems and has a broader (beyond earth) scope, the registry defined below is meant to be more easily modified.

TODO: Open question, should we create a new registry here or attempt to modify the one created by [RFC5870]. It's worth noting that we include the ability to specify any geodetic system including ones designed for astronomical bodies other than the earth, as well as ones based on alternate systems. These requirements may be too broad for adapting the existing [RFC5870] registry.

TODO: Open question, is FCFS too easy, perhaps expert review would strike a good balance. If expert review is acceptable, would it also be acceptable to update the policy on [RFC5870] and use it instead?

The allocation policy for this registry is First Come First Served, [RFC8126] as the intent is simply to avoid duplicate values.

The initial values for this registry are as follows.

Name	Description
me	Mean Earth/Polar Axis (Moon)
mola-vik-1	MOLA Height, IAU Viking-1 PM (Mars)
wgs-84-96	World Geodetic System 1984 [WGS84] w/ EGM96
wgs-84-08	World Geodetic System 1984 [WGS84] w/ [EGM08]
wgs-84	World Geodetic System 1984 [WGS84] (EGM96 or better)

7. Security Considerations

This document defines a common geo location grouping using the YANG data modeling language. The grouping itself has no security or privacy impact on the Internet, but the usage of the grouping in concrete YANG modules might have. The security considerations

spelled out in the YANG 1.1 specification [RFC7950] apply for this document as well.

8. References

8.1. Normative References

[EGM08] Pavlis, N., Holmes, S., Kenyon, S., and J. Factor, "An Earth Gravitational Model to Degree 2160: EGM08.", 2008, <http://earth-info.nga.mil/GandG/wgs84/gravitymod/egm2008/egm08_wgs84.html>.

[EGM96] Lemoine, F., Kenyon, S., Factor, J., Trimmer, R., Pavlis, N., Chinn, D., Cox, C., Klosko, S., Luthcke, S., Torrence, M., Wang, Y., Williamson, R., Pavlis, E., Rapp, R., and T. Olson, "The Development of the Joint NASA GSFC and the National Imagery and Mapping Agency (NIMA) Geopotential Model EGM96.", 1998, <https://cddis.nasa.gov/926/egm96/egm96.html>.

[ISO.6709.2008] International Organization for Standardization, "ISO 6709:2008 Standard representation of geographic point location by coordinates.", 2008.

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

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[WGS84] National Imagery and Mapping Agency., "National Imagery and Mapping Agency Technical Report 8350.2, Third Edition.", 1 2000, <http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf>.

8.2. Informative References

[ISO.19136.2007]
International Organization for Standardization, "ISO 19136:2007 Geographic information -- Geography Markup Language (GML)".

[KML22] Wilson, T., Ed., "OGC KML (Version 2.2)", 4 2008, <http://portal.opengeospatial.org/files/?artifact_id=27810>.

[KML23] Burggraf, D., Ed., "OGC KML 2.3", 8 2015, <http://docs.opengeospatial.org/is/12-007r2/12-007r2.html>.

[RFC5870] Mayrhofer, A. and C. Spanring, "A Uniform Resource Identifier for Geographic Locations ('geo' URI)", RFC 5870, DOI 10.17487/RFC5870, June 2010, <https://www.rfc-editor.org/info/rfc5870>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[W3CGEO] Popescu, A., "Geolocation API Specification", 11 2016, <https://www.w3.org/TR/2016/REC-geolocation-API-20161108/>.

Appendix A. Examples

Below is a fictitious module that uses the geo-location grouping.

```
module example-uses-geo-location {
  namespace
    "urn:example:example-uses-geo-location";
  prefix ugeo;
  import ietf-geo-location { prefix geo; }
  organization "Empty Org";
  contact "Example Author <eauthor@example.com>";
  description "Example use of geo-location";
  revision 2019-02-02 { reference "None"; }
  container locatable-items {
    description "container of locatable items";
    list locatable-item {
      key name;
      description "A of locatable item";
      leaf name {
        type string;
        description "name of locatable item";
      }
      uses geo:geo-location;
    }
  }
}
```

Figure 3: Example YANG module using geo location.

Below is a the YANG tree for the fictitious module that uses the geo-location grouping.

```
module: example-uses-geo-location
  +--rw locatable-items
     +--rw locatable-item* [name]
        +--rw name            string
        +--rw geo-location
           +--rw reference-frame
           |  +--rw alternate-system?    string {alternate-systems}?
           |  +--rw astronomical-body?   string
           |  +--rw geodetic-system
           |     +--rw geodetic-datum?    string
           |     +--rw coord-accuracy?    decimal64
           |     +--rw height-accuracy?   decimal64
           +--rw (location)?
           |  +--:(ellipsoid)
           |  |  +--rw latitude?    degrees
           |  |  +--rw longitude?   degrees
           |  |  +--rw height?      decimal64
           |  +--:(cartesian)
           |     +--rw x?           decimal64
           |     +--rw y?           decimal64
           |     +--rw z?           decimal64
           +--rw velocity
           |  +--rw v-north?   decimal64
           |  +--rw v-east?    decimal64
           |  +--rw v-up?      decimal64
           +--rw timestamp?         types:date-and-time
```

Below is some example YANG XML data for the fictitious module that uses the geo-location grouping.

```
<ns0:config xmlns:ns0="urn:ietf:params:xml:ns:netconf:base:1.0">
  <locatable-items xmlns="urn:example:example-uses-geo-location">
    <locatable-item>
      <name>Gaetana's</name>
      <geo-location>
        <latitude>40.73297</latitude>
        <longitude>-74.007696</longitude>
      </geo-location>
    </locatable-item>
    <locatable-item>
      <name>Pont des Arts</name>
      <geo-location>
        <timestamp>2012-03-31T16:00:00Z</timestamp>
        <latitude>48.8583424</latitude>
        <longitude>2.3375084</longitude>
        <height>35</height>
      </geo-location>
    </locatable-item>
```

```
      <locatable-item>
        <name>Saint Louis Cathedral</name>
        <geo-location>
          <timestamp>2013-10-12T15:00:00-06:00</timestamp>
          <latitude>29.9579735</latitude>
          <longitude>-90.0637281</longitude>
        </geo-location>
      </locatable-item>
      <locatable-item>
        <name>Apollo 11 Landing Site</name>
        <geo-location>
          <timestamp>1969-07-21T02:56:15Z</timestamp>
          <reference-frame>
            <astronomical-body>moon</astronomical-body>
            <geodetic-system>
              <geodetic-datum>me</geodetic-datum>
            </geodetic-system>
          </reference-frame>
          <latitude>0.67409</latitude>
          <longitude>23.47298</longitude>
        </geo-location>
      </locatable-item>
      <locatable-item>
        <name>Reference Frame Only</name>
        <geo-location>
          <reference-frame>
            <astronomical-body>moon</astronomical-body>
            <geodetic-system>
              <geodetic-datum>me</geodetic-datum>
            </geodetic-system>
          </reference-frame>
        </geo-location>
      </locatable-item>
    </locatable-items>
  </ns0:config>
```

Figure 4: Example XML data of geo location use.

Appendix B. Acknowledgements

We would like to thank Peter Lothberg for the motivation as well as help in defining a more broadly useful geographic location object.

We would also like to thank Acee Lindem and Qin Wu for their work on a geographic location object that led to this documents creation.

Author's Address

Christian Hopps
LabN Consulting, L.L.C.

Email: chopps@chopps.org

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2020

R. Wilton, Ed.
D. Ball
T. Singh
Cisco Systems
S. Sivaraj
Juniper Networks

November 4, 2019

Common Interface Extension YANG Data Models
draft-ietf-netmod-intf-ext-yang-08

Abstract

This document defines two YANG modules that augment the Interfaces data model defined in the "YANG Data Model for Interface Management" with additional configuration and operational data nodes to support common lower layer interface properties, such as interface MTU.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) defined in RFC 8342.

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 https://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 7, 2020.

Copyright Notice

Copyright (c) 2019 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 (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents

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

Table of Contents

1. Introduction 3
1.1. Terminology 3
1.2. Tree Diagrams 4
2. Interface Extensions Module 4
2.1. Carrier Delay 5
2.2. Dampening 6
2.2.1. Suppress Threshold 7
2.2.2. Half-Life Period 7
2.2.3. Reuse Threshold 7
2.2.4. Maximum Suppress Time 7
2.3. Encapsulation 7
2.4. Loopback 8
2.5. Maximum frame size 8
2.6. Sub-interface 8
2.7. Forwarding Mode 9
3. Interfaces Ethernet-Like Module 9
4. Interface Extensions YANG Module 10
5. Interfaces Ethernet-Like YANG Module 20
6. Examples 24
6.1. Carrier delay configuration 24
6.2. Dampening configuration 25
6.3. MAC address configuration 26
7. Acknowledgements 28
8. ChangeLog 28
8.1. Version -08 28
8.2. Version -07 28
8.3. Version -06 28
8.4. Version -05 28
8.5. Version -04 28
8.6. Version -03 28
8.7. Version -02 28
9. IANA Considerations 29
10. Security Considerations 29
10.1. ietf-if-extensions.yang 29
10.2. ietf-if-ethernet-like.yang 30
11. References 30
11.1. Normative References 30
11.2. Informative References 31
Authors' Addresses 32

1. Introduction

This document defines two NMDA compatible [RFC8342] YANG 1.1 [RFC7950] modules for the management of network interfaces. It defines various augmentations to the generic interfaces data model [RFC8343] to support configuration of lower layer interface properties that are common across many types of network interface.

One of the aims of this document is to provide a standard definition for these configuration items regardless of the underlying interface type. For example, a definition for configuring or reading the MAC address associated with an interface is provided that can be used for any interface type that uses Ethernet framing.

Several of the augmentations defined here are not backed by any formal standard specification. Instead, they are for features that are commonly implemented in equivalent ways by multiple independent network equipment vendors. The aim of this document is to define common paths and leaves for the configuration of these equivalent features in a uniform way, making it easier for users of the YANG model to access these features in a vendor independent way. Where necessary, a description of the expected behavior is also provided with the aim of ensuring vendors implementations are consistent with the specified behaviour.

Given that the modules contain a collection of discrete features with the common theme that they generically apply to interfaces, it is plausible that not all implementors of the YANG module will decide to support all features. Hence separate feature keywords are defined for each logically discrete feature to allow implementors the flexibility to choose which specific parts of the model they support.

The augmentations are split into two separate YANG modules that each focus on a particular area of functionality. The two YANG modules defined in this document are:

ietf-if-extensions.yang - Defines extensions to the IETF interface data model to support common configuration data nodes.

ietf-if-ethernet-like.yang - Defines a module for any configuration and operational data nodes that are common across interfaces that use Ethernet framing.

1.1. Terminology

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

14 RFC 2119 [RFC2119] RFC 8174 [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

2. Interface Extensions Module

The Interfaces Extensions YANG module provides some basic extensions to the IETF interfaces YANG module.

The module provides:

- A carrier delay feature used to provide control over short lived link state flaps.
- An interface link state dampening feature that is used to provide control over longer lived link state flaps.
- An encapsulation container and extensible choice statement for use by any interface types that allow for configurable L2 encapsulations.
- A loopback configuration leaf that is primarily aimed at loopback at the physical layer.
- MTU configuration leaves applicable to all packet/frame based interfaces.
- A forwarding mode leaf to indicate the OSI layer at which the interface handles traffic.
- A generic "sub-interface" identity that an interface identity definition can derive from if it defines a sub-interface.
- A parent interface leaf useable for all types of sub-interface that are children of parent interfaces.

The "ietf-if-extensions" YANG module has the following structure:

```
module: ietf-if-extensions
  augment /if:interfaces/if:interface:
    +--rw carrier-delay {carrier-delay}?
    |  +--rw down?                  uint32
    |  +--rw up?                    uint32
    |  +--ro carrier-transitions?   yang:counter64
    |  +--ro timer-running?         enumeration
    +--rw dampening! {dampening}?
    |  +--rw half-life?           uint32
    |  +--rw reuse?               uint32
    |  +--rw suppress?            uint32
    |  +--rw max-suppress-time?   uint32
    |  +--ro penalty?             uint32
    |  +--ro suppressed?          boolean
    |  +--ro time-remaining?      uint32
    +--rw encapsulation
    |  +--rw (encaps-type)?
    +--rw loopback?          identityref {loopback}?
    +--rw max-frame-size?    uint32 {max-frame-size}?
    +--ro forwarding-mode?   identityref
  augment /if:interfaces/if:interface:
    +--rw parent-interface    if:interface-ref {sub-interfaces}?
```

2.1. Carrier Delay

The carrier delay feature augments the IETF interfaces data model with configuration for a simple algorithm that is used, generally on physical interfaces, to suppress short transient changes in the interface link state. It can be used in conjunction with the dampening feature described in Section 2.2 to provide effective control of unstable links and unwanted state transitions.

The principle of the carrier delay feature is to use a short per interface timer to ensure that any interface link state transition that occurs and reverts back within the specified time interval is entirely suppressed without providing any signalling to any upper layer protocols that the state transition has occurred. E.g. in the case that the link state transition is suppressed then there is no change of the /if:interfaces/if:interface/oper-status or /if:interfaces/if:interfaces/last-change leaves for the interface that the feature is operating on. One obvious side effect of using this feature that is that any state transition will always be delayed by the specified time interval.

The configuration allows for separate timer values to be used in the suppression of down->up->down link transitions vs up->down->up link transitions.

The carrier delay down timer leaf specifies the amount of time that an interface that is currently in link up state must be continuously down before the down state change is reported to higher level protocols. Use of this timer can cause traffic to be black holed for the configured value and delay reconvergence after link failures, therefore its use is normally restricted to cases where it is necessary to allow enough time for another protection mechanism (such as an optical layer automatic protection system) to take effect.

The carrier delay up timer leaf specifies the amount of time that an interface that is currently in link down state must be continuously up before the down->up link state transition is reported to higher level protocols. This timer is generally useful as a debounce mechanism to ensure that a link is relatively stable before being brought into service. It can also be used effectively to limit the frequency at which link state transition events may occur. The default value for this leaf is determined by the underlying network device.

2.2. Dampening

The dampening feature introduces a configurable exponential decay mechanism to suppress the effects of excessive interface link state flapping. This feature allows the network operator to configure a device to automatically identify and selectively dampen a local interface which is flapping. Dampening an interface keeps the interface operationally down until the interface stops flapping and becomes stable. Configuring the dampening feature can improve convergence times and stability throughout the network by isolating failures so that disturbances are not propagated, which reduces the utilization of system processing resources by other devices in the network and improves overall network stability.

The basic algorithm uses a counter that is increased by 1000 units every time the underlying interface link state changes from up to down. If the counter increases above the suppress threshold then the interface is kept down (and out of service) until either the maximum suppression time is reached, or the counter has reduced below the reuse threshold. The half-life period determines that rate at which the counter is periodically reduced by half.

2.2.1. Suppress Threshold

The suppress threshold is the value of the accumulated penalty that triggers the device to dampen a flapping interface. The flapping interface is identified by the device and assigned a penalty for each up to down link state change, but the interface is not automatically dampened. The device tracks the penalties that a flapping interface accumulates. When the accumulated penalty reaches or exceeds the suppress threshold, the interface is placed in a suppressed state.

2.2.2. Half-Life Period

The half-life period determines how fast the accumulated penalties can decay exponentially. The accumulated penalty decays at a rate that causes its value to be reduced by half after each half-life period.

2.2.3. Reuse Threshold

If, after one or more half-life periods, the accumulated penalty decreases below the reuse threshold and the underlying interface link state is up then the interface is taken out of suppressed state and is allowed to go up.

2.2.4. Maximum Suppress Time

The maximum suppress time represents the maximum amount of time an interface can remain dampened when a new penalty is assigned to an interface. The default of the maximum suppress timer is four times the half-life period. The maximum value of the accumulated penalty is calculated using the maximum suppress time, reuse threshold and half-life period.

2.3. Encapsulation

The encapsulation container holds a choice node that is to be augmented with datalink layer specific encapsulations, such as HDLC, PPP, or sub-interface 802.1Q tag match encapsulations. The use of a choice statement ensures that an interface can only have a single datalink layer protocol configured.

The different encapsulations themselves are defined in separate YANG modules defined in other documents that augument the encapsulation choice statement. For example the Ethernet specific basic 'dot1q-vlan' encapsulation is defined in ietf-if-l3-vlan.yang and the 'flexible' encapsulation is defined in ietf-flexible-encapsulation.yang, both modules from [I-D.ietf-netmod-sub-intf-vlan-model].

2.4. Loopback

The loopback configuration leaf allows any physical interface to be configured to be in one of the possible following physical loopback modes, i.e. internal loopback, line loopback, or use of an external loopback connector. The use of YANG identities allows for the model to be extended with other modes of loopback if required.

The following loopback modes are defined:

- Internal loopback - All egress traffic on the interface is internally looped back within the interface to be received on the ingress path.

- Line loopback - All ingress traffic received on the interface is internally looped back within the interface to the egress path.

- Loopback Connector - The interface has a physical loopback connector attached that loops all egress traffic back into the interface's ingress path, with equivalent semantics to internal loopback.

2.5. Maximum frame size

A maximum frame size configuration leaf (max-frame-size) is provided to specify the maximum size of a layer 2 frame that may be transmitted or received on an interface. The value includes the overhead of any layer 2 header, the maximum length of the payload, and any frame check sequence (FCS) bytes. If configured, the max-frame-size leaf on an interface also restricts the max-frame-size of any child sub-interfaces, and the available MTU for protocols.

2.6. Sub-interface

The sub-interface feature specifies the minimal leaves required to define a child interface that is parented to another interface.

A sub-interface is a logical interface that handles a subset of the traffic on the parent interface. Separate configuration leaves are used to classify the subset of ingress traffic received on the parent interface to be processed in the context of a given sub-interface. All egress traffic processed on a sub-interface is given to the parent interface for transmission. Otherwise, a sub-interface is like any other interface in /if:interfaces and supports the standard interface features and configuration.

For some vendor specific interface naming conventions the name of the child interface is sufficient to determine the parent interface,

which implies that the child interface can never be reparented to a different parent interface after it has been created without deleting the existing sub-interface and recreating a new sub-interface. Even in this case it is useful to have a well defined leaf to cleanly identify the parent interface.

The model also allows for arbitrarily named sub-interfaces by having an explicit parent-interface leaf define the child -> parent relationship. In this naming scenario it is also possible for implementations to allow for logical interfaces to be reparented to new parent interfaces without needing the sub-interface to be destroyed and recreated.

2.7. Forwarding Mode

The forwarding mode leaf provides additional information as to what mode or layer an interface is logically operating and forwarding traffic at. The implication of this leaf is that for traffic forwarded at a given layer that any headers for lower layers are stripped off before the packet is forwarded at the given layer. Conversely, on egress any lower layer headers must be added to the packet before it is transmitted out of the interface.

The following forwarding modes are defined:

- Physical - Traffic is being forwarded at the physical layer. This includes DWDM or OTN based switching.
- Data-link - Layer 2 based forwarding, such as Ethernet/VLAN based switching, or L2VPN services.
- Network - Network layer based forwarding, such as IP, MPLS, or L3VPNs.

3. Interfaces Ethernet-Like Module

The Interfaces Ethernet-Like Module is a small module that contains all configuration and operational data that is common across interface types that use Ethernet framing as their datalink layer encapsulation.

This module currently contains leaves for the configuration and reporting of the operational MAC address and the burnt-in MAC address (BIA) associated with any interface using Ethernet framing.

The "ietf-if-ethernet-like" YANG module has the following structure:

```
module: ietf-if-ethernet-like
  augment /if:interfaces/if:interface:
    +--rw ethernet-like
       +--rw mac-address?       yang:mac-address
       |       {configurable-mac-address}?
       +--ro bia-mac-address?   yang:mac-address
  augment /if:interfaces/if:interface/if:statistics:
    +--ro in-drop-unknown-dest-mac-pkts?   yang:counter64
```

4. Interface Extensions YANG Module

This YANG module augments the interface container defined in RFC 8343 [RFC8343].

```
<CODE BEGINS> file "ietf-if-extensions@2019-11-04.yang"
module ietf-if-extensions {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-if-extensions";

  prefix if-ext;

  import ietf-yang-types {
    prefix yang;
    reference "RFC 6991: Common YANG Data Types";
  }

  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model For Interface Management";
  }

  import iana-if-type {
    prefix ianaift;
    reference "RFC 7224: IANA Interface Type YANG Module";
  }

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

  contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
```

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

        Editor:   Robert Wilton
                  <mailto:rwilton@cisco.com>";

    description
      "This module contains common definitions for extending the IETF
       interface YANG model (RFC 8343) with common configurable layer 2
       properties.

       Copyright (c) 2019 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
       (https://trustee.ietf.org/license-info).

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

       The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
       NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
       'MAY', and 'OPTIONAL' in this document are to be interpreted as
       described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
       they appear in all capitals, as shown here.";

    revision 2019-11-04 {
      description
        "Initial revision.";

      reference
        "RFC XXX, Common Interface Extension YANG Data Models";
    }

    feature carrier-delay {
      description
        "This feature indicates that configurable interface
         carrier delay is supported, which is a feature is used to
         limit the propagation of very short interface link state
         flaps.";
      reference "RFC XXX, Section 2.1 Carrier Delay";
    }

    feature dampening {
```

```
    description
      "This feature indicates that the device supports interface
       dampening, which is a feature that is used to limit the
       propagation of interface link state flaps over longer
       periods.";
    reference "RFC XXX, Section 2.2 Dampening";
  }

  feature loopback {
    description
      "This feature indicates that configurable interface loopback
       is supported.";
    reference "RFC XXX, Section 2.4 Loopback";
  }

  feature max-frame-size {
    description
      "This feature indicates that the device supports configuring
       or reporting the maximum frame size on interfaces.";
    reference "RFC XXX, Section 2.5 Maximum Frame Size";
  }

  feature sub-interfaces {
    description
      "This feature indicates that the device supports the
       instantiation of sub-interfaces.  Sub-interfaces are defined
       as logical child interfaces that allow features and forwarding
       decisions to be applied to a subset of the traffic processed
       on the specified parent interface.";
    reference "RFC XXX, Section 2.6 Sub-interface";
  }

  /*
   * Define common identities to help allow interface types to be
   * assigned properties.
   */
  identity sub-interface {
    description
      "Base type for generic sub-interfaces.

       New or custom interface types can derive from this type to
       inherit generic sub-interface configuration.";
    reference "RFC XXX, Section 2.6 Sub-interface";
  }

  identity ethSubInterface{
    base ianaift:l2vlan;
    base sub-interface;
```

```
      description
        "This identity represents the child sub-interface of any
         interface types that uses Ethernet framing (with or without
         802.1Q tagging).";
    }

    identity loopback {
      description "Base identity for interface loopback options";
      reference "RFC XXX, Section 2.4";
    }
    identity internal {
      base loopback;
      description
        "All egress traffic on the interface is internally looped back
         within the interface to be received on the ingress path.";
      reference "RFC XXX, Section 2.4";
    }
    identity line {
      base loopback;
      description
        "All ingress traffic received on the interface is internally
         looped back within the interface to the egress path.";
      reference "RFC XXX, Section 2.4";
    }
    identity connector {
      base loopback;
      description
        "The interface has a physical loopback connector attached that
         loops all egress traffic back into the interface's ingress
         path, with equivalent semantics to loopback internal.";
      reference "RFC XXX, Section 2.4";
    }


    identity forwarding-mode {
      description "Base identity for forwarding-mode options.";
      reference "RFC XXX, Section 2.7";
    }
    identity physical {
      base forwarding-mode;
      description
        "Physical layer forwarding.  This includes DWDM or OTN based
         optical switching.";
      reference "RFC XXX, Section 2.7";
    }
    identity data-link {
      base forwarding-mode;
      description
```

```
        "Layer 2 based forwarding, such as Ethernet/VLAN based
         switching, or L2VPN services.";
      reference "RFC XXX, Section 2.7";
    }
    identity network {
      base forwarding-mode;
      description
        "Network layer based forwarding, such as IP, MPLS, or L3VPNs.";
      reference "RFC XXX, Section 2.7";
    }


    /*
     * Augments the IETF interfaces model with leaves to configure
     * and monitor carrier-delay on an interface.
     */
    augment "/if:interfaces/if:interface" {
      description
        "Augments the IETF interface model with optional common
         interface level commands that are not formally covered by any
         specific standard.";

      /*
       * Defines standard YANG for the Carrier Delay feature.
       */
      container carrier-delay {
        if-feature "carrier-delay";
        description
          "Holds carrier delay related feature configuration.";
        leaf down {
          type uint32;
          units milliseconds;
          description
            "Delays the propagation of a 'loss of carrier signal' event
             that would cause the interface state to go down, i.e. the
             command allows short link flaps to be suppressed. The
             configured value indicates the minimum time interval (in
             milliseconds) that the carrier signal must be continuously
             down before the interface state is brought down. If not
             configured, the behaviour on loss of carrier signal is
             vendor/interface specific, but with the general
             expectation that there should be little or no delay.";
        }
        leaf up {
          type uint32;
          units milliseconds;
          description
            "Defines the minimum time interval (in milliseconds) that
```

```
            the carrier signal must be continuously present and error
            free before the interface state is allowed to transition
            from down to up.  If not configured, the behaviour is
            vendor/interface specific, but with the general
            expectation that sufficient default delay should be used
            to ensure that the interface is stable when enabled before
            being reported as being up.  Configured values that are
            too low for the hardware capabilties may be rejected.";
        }
        leaf carrier-transitions {
          type yang:counter64;
          units transitions;
          config false;
          description
            "Defines the number of times the underlying carrier state
             has changed to, or from, state up.  This counter should be
             incremented even if the high layer interface state changes
             are being suppressed by a running carrier-delay timer.";
        }
        leaf timer-running {
          type enumeration {
            enum none {
              description
                "No carrier delay timer is running.";
            }
            enum up {
              description
                "Carrier-delay up timer is running.  The underlying
                 carrier state is up, but interface state is not
                 reported as up.";
            }
            enum down {
              description
                "Carrier-delay down timer is running.  Interface state
                 is reported as up, but the underlying carrier state is
                 actually down.";
            }
          }
          config false;
          description
            "Reports whether a carrier delay timer is actively running,
             in which case the interface state does not match the
             underlying carrier state.";
        }

        reference "RFC XXX, Section 2.1 Carrier Delay";
      }
```

```
      /*
       * Augments the IETF interfaces model with a container to hold
       * generic interface dampening
       */
      container dampening {
        if-feature "dampening";
        presence
          "Enable interface link flap dampening with default settings
           (that are vendor/device specific).";
        description
          "Interface dampening limits the propagation of interface link
           state flaps over longer periods.";
        reference "RFC XXX, Section 2.2 Dampening";

        leaf half-life {
          type uint32;
          units seconds;
          description
            "The time (in seconds) after which a penalty would be half
             its original value.  Once the interface has been assigned
             a penalty, the penalty is decreased at a decay rate
             equivalent to the half-life.  For some devices, the
             allowed values may be restricted to particular multiples
             of seconds.  The default value is vendor/device
             specific.";
          reference "RFC XXX, Section 2.3.2 Half-Life Period";
        }

        leaf reuse {
          type uint32;
          description
            "Penalty value below which a stable interface is
             unsuppressed (i.e. brought up) (no units).  The default
             value is vendor/device specific.  The penalty value for a
             link up->down state change is 1000 units.";
          reference "RFC XXX, Section 2.2.3 Reuse Threshold";
        }

        leaf suppress {
          type uint32;
          description
            "Limit at which an interface is suppressed (i.e. held down)
             when its penalty exceeds that limit (no units).  The value
             must be greater than the reuse threshold.  The default
             value is vendor/device specific.  The penalty value for a
             link up->down state change is 1000 units.";
          reference "RFC XXX, Section 2.2.1 Suppress Threshold";
        }
```

```
        leaf max-suppress-time {
          type uint32;
          units seconds;
          description
            "Maximum time (in seconds) that an interface can be
             suppressed before being unsuppressed if no further link
             up->down state change penalties have been applied.  This
             value effectively acts as a ceiling that the penalty value
             cannot exceed.  The default value is vendor/device
             specific.";
          reference "RFC XXX, Section 2.2.4 Maximum Suppress Time";
        }

        leaf penalty {
          type uint32;
          config false;
          description
            "The current penalty value for this interface.  When the
             penalty value exceeds the 'suppress' leaf then the
             interface is suppressed (i.e. held down).";
          reference "RFC XXX, Section 2.2 Dampening";
        }

        leaf suppressed {
          type boolean;
          config false;
          description
            "Represents whether the interface is suppressed (i.e. held
             down) because the 'penalty' leaf value exceeds the
             'suppress' leaf.";
          reference "RFC XXX, Section 2.2 Dampening";
        }

        leaf time-remaining {
          when '../suppressed = "true"' {
            description
              "Only suppressed interfaces have a time remaining.";
          }
          type uint32;
          units seconds;
          config false;
          description
            "For a suppressed interface, this leaf represents how long
             (in seconds) that the interface will remain suppressed
             before it is allowed to go back up again.";
          reference "RFC XXX, Section 2.2 Dampening";
        }
      }
```

```
      /*
       * Various types of interfaces support a configurable layer 2
       * encapsulation, any that are supported by YANG should be
       * listed here.
       *
       * Different encapsulations can hook into the common encaps-type
       * choice statement.
       */
      container encapsulation {
        when
          "derived-from-or-self(../if:type,
                                'ianaift:ethernetCsmacd') or
           derived-from-or-self(../if:type,
                                'ianaift:ieee8023adLag') or
           derived-from-or-self(../if:type, 'ianaift:pos') or
           derived-from-or-self(../if:type,
                                'ianaift:atmSubInterface') or
           derived-from-or-self(../if:type, 'ethSubInterface')" {

          description
            "All interface types that can have a configurable L2
             encapsulation.";
        }

        description
          "Holds the OSI layer 2 encapsulation associated with an
           interface.";
        choice encaps-type {
          description
            "Extensible choice of layer 2 encapsulations";
          reference "RFC XXX, Section 2.3 Encapsulation";
        }
      }

       /*
        * Various types of interfaces support loopback configuration,
        * any that are supported by YANG should be listed here.
        */
      leaf loopback {
        when "derived-from-or-self(../if:type,
                                   'ianaift:ethernetCsmacd') or
              derived-from-or-self(../if:type, 'ianaift:sonet') or
              derived-from-or-self(../if:type, 'ianaift:atm') or
              derived-from-or-self(../if:type, 'ianaift:otnOtu')" {
          description
            "All interface types that support loopback configuration.";
        }
        if-feature "loopback";
```

```
        type identityref {
          base loopback;
        }
        description "Enables traffic loopback.";
        reference "RFC XXX, Section 2.4 Loopback";
      }

      /*
       * Allows the maximum frame size to be configured or reported.
       */
      leaf max-frame-size {
        if-feature "max-frame-size";
        type uint32 {
          range "64 .. max";
        }
        description
          "The maximum size of layer 2 frames that may be transmitted
           or received on the interface (including any frame header,
           maximum frame payload size, and frame checksum sequence).

           If configured, the max-frame-size also limits the maximum
           frame size of any child sub-interfaces.  The MTU available
           to higher layer protocols is restricted to the maximum frame
           payload size, and MAY be further restricted by explicit
           layer 3 or protocol specific MTU configuration.";

        reference "RFC XXX, Section 2.5 Maximum Frame Size";
      }

      /*
       * Augments the IETF interfaces model with a leaf that indicates
       * which mode, or layer, is being used to forward the traffic.
       */
      leaf forwarding-mode {
        type identityref {
          base forwarding-mode;
        }
        config false;

        description
          "The forwarding mode that the interface is operating in.";
        reference "RFC XXX, Section 2.7 Forwarding Mode";
      }
    }

    /*
     * Add generic support for sub-interfaces.
     *
```

```
      * This should be extended to cover all interface types that are
      * child interfaces of other interfaces.
      */
     augment "/if:interfaces/if:interface" {
       when "derived-from(if:type, 'sub-interface') or
             derived-from-or-self(if:type, 'ianaift:atmSubInterface') or
             derived-from-or-self(if:type, 'ianaift:frameRelay')"  {
         description
           "Any ianaift:types that explicitly represent sub-interfaces
            or any types that derive from the sub-interface identity.";
       }
       if-feature "sub-interfaces";

       description
         "Adds a parent interface field to interfaces that model
          sub-interfaces.";
       leaf parent-interface {

         type if:interface-ref;

         mandatory true;
         description
           "This is the reference to the parent interface of this
            sub-interface.";
         reference "RFC XXX, Section 2.6 Sub-interface";
       }
     }
   }
   <CODE ENDS>
```

5. Interfaces Ethernet-Like YANG Module

This YANG module augments the interface container defined in RFC 8343 [RFC8343] for Ethernet-like interfaces. This includes Ethernet interfaces, 802.3 LAG (802.1AX) interfaces, VLAN sub-interfaces, Switch Virtual interfaces, and Pseudo-Wire Head-End interfaces.

```
  <CODE BEGINS> file "ietf-if-ethernet-like@2019-11-04.yang"
  module ietf-if-ethernet-like {
    yang-version 1.1;

    namespace
      "urn:ietf:params:xml:ns:yang:ietf-if-ethernet-like";

    prefix ethlike;
```

```
  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model For Interface Management";
  }

  import ietf-yang-types {
    prefix yang;
    reference "RFC 6991: Common YANG Data Types";
  }

  import iana-if-type {
    prefix ianaift;
    reference "RFC 7224: IANA Interface Type YANG Module";
  }

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

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

     Editor:   Robert Wilton
               <mailto:rwilton@cisco.com>";

  description
    "This module contains YANG definitions for configuration for
     'Ethernet-like' interfaces.  It is applicable to all interface
     types that use Ethernet framing and expose an Ethernet MAC
     layer, and includes such interfaces as physical Ethernet
     interfaces, Ethernet LAG interfaces and VLAN sub-interfaces.

     Additional interface configuration and counters for physical
     Ethernet interfaces are defined in
     ieee802-ethernet-interface.yang, as part of IEEE Std
     802.3.2-2019.

     Copyright (c) 2019 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
     (https://trustee.ietf.org/license-info).
```

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

    revision 2019-11-04 {
      description "Initial revision.";

      reference
        "RFC XXX, Common Interface Extension YANG Data Models";
    }

    feature configurable-mac-address {
      description
        "This feature indicates that MAC addresses on Ethernet-like
         interfaces can be configured.";
      reference "RFC XXX, Section 3 Interfaces Ethernet-Like Module";
    }


    /*
     * Configuration parameters for Ethernet-like interfaces.
     */
    augment "/if:interfaces/if:interface" {
      when "derived-from-or-self(if:type, 'ianaift:ethernetCsmacd') or
            derived-from-or-self(if:type, 'ianaift:ieee8023adLag') or
            derived-from-or-self(if:type, 'ianaift:ifPwType')" {
        description "Applies to all Ethernet-like interfaces";
      }
      description
        "Augment the interface model with parameters for all
         Ethernet-like interfaces.";

      container ethernet-like {
        description
          "Contains parameters for interfaces that use Ethernet framing
           and expose an Ethernet MAC layer.";

        leaf mac-address {
          if-feature "configurable-mac-address";
          type yang:mac-address;
          description
            "The MAC address of the interface.  The operational value
             matches the /if:interfaces/if:interface/if:phys-address
             leaf defined in ietf-interface.yang.";
        }

        leaf bia-mac-address {
          type yang:mac-address;
```

```
          config false;
          description
            "The 'burnt-in' MAC address.  I.e the default MAC address
             assigned to the interface if no MAC address has been
             explicitly configured on it.";
        }
      }
    }


    /*
     * Configuration parameters for Ethernet-like interfaces.
     */
    augment "/if:interfaces/if:interface/if:statistics" {
      when "derived-from-or-self(../if:type,
                                 'ianaift:ethernetCsmacd') or
            derived-from-or-self(../if:type,
                                 'ianaift:ieee8023adLag') or
            derived-from-or-self(../if:type, 'ianaift:ifPwType')" {
        description "Applies to all Ethernet-like interfaces";
      }
      description
        "Augment the interface model statistics with additional
         counters related to Ethernet-like interfaces.";

      leaf in-drop-unknown-dest-mac-pkts {
        type yang:counter64;
        units frames;
        description
          "A count of the number of frames that were well formed, but
           otherwise dropped because the destination MAC address did
           not pass any ingress destination MAC address filter.

           For consistency, frames counted against this drop counters
           are also counted against the IETF interfaces statistics.  In
           particular, they are included in in-octets and in-discards,
           but are not included in in-unicast-pkts, in-multicast-pkts
           or in-broadcast-pkts, because they are not delivered to a
           higher layer.

           Discontinuities in the values of this counter can occur at
           re-initialization of the management system, and at other
           times as indicated by the value of the 'discontinuity-time'
           leaf defined in the ietf-interfaces YANG module (RFC 8343).";
      }
    }
  }
  <CODE ENDS>
```

6. Examples

The following sections give some examples of how different parts of the YANG modules could be used. Examples are not given for the more trivial configuration, or for sub-interfaces, for which examples are contained in [I-D.ietf-netmod-sub-intf-vlan-model].

6.1. Carrier delay configuration

The following example shows how the operational state datastore could look like for an Ethernet interface without any carrier delay configuration. The down leaf value of 0 indicates that link down events as always propagated to high layers immediately, but an up leaf value of 50 indicates that the interface must be up and stable for at least 50 msecs before the interface is reported as being up to the high layers.

```
<?xml version="1.0" encoding="utf-8"?>
<interfaces
 xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
 xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
xmlns:if-ext="urn:ietf:params:xml:ns:yang:ietf-if-extensions">
  <interface>
    <name>eth0</name>
    <type>ianaift:ethernetCsmacd</type>
    <if-ext:carrier-delay>
      <if-ext:down>0</if-ext:down>
      <if-ext:up>50</if-ext:up>
    </if-ext:carrier-delay>
  </interface>
</interfaces>
```

The following example shows explicit carrier delay up and down values have been configured. A 50 msec down leaf value has been used to potentially allow optical protection to recover the link before the higher layer protocol state is flapped. A 1 second (1000 milliseconds) up leaf value has been used to ensure that the link is always reasonably stable before allowing traffic to be carried over it. This also has the benefit of greatly reducing the rate at which higher layer protocol state flaps could occur.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces
    xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
    xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
    xmlns:if-ext="urn:ietf:params:xml:ns:yang:ietf-if-extensions">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
      <if-ext:carrier-delay>
        <if-ext:down>50</if-ext:down>
        <if-ext:up>1000</if-ext:up>
      </if-ext:carrier-delay>
    </interface>
  </interfaces>
</config>
```

6.2. Dampening configuration

The following example shows what the operational state datastore may look like for an interface configured with interface dampening. The 'suppressed' leaf indicates that the interface is currently suppressed (i.e. down) because the 'penalty' is greater than the 'suppress' leaf threshold. The 'time-remaining' leaf indicates that the interface will remain suppressed for another 103 seconds before the 'penalty' is below the 'reuse' leaf value and the interface is allowed to go back up again.

```
<?xml version="1.0" encoding="utf-8"?>
<interfaces
 xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
 xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
  <interface>
    <name>eth0</name>
    <type>ianaift:ethernetCsmacd</type>
    <oper-status>down</oper-status>
    <dampening
     xmlns="urn:ietf:params:xml:ns:yang:ietf-if-extensions">
      <half-life>60</half-life>
      <reuse>750</reuse>
      <suppress>2000</suppress>
      <max-suppress-time>240</max-suppress-time>
      <penalty>2480</penalty>
      <suppressed>true</suppressed>
      <time-remaining>103</time-remaining>
    </dampening>
  </interface>
</interfaces>
```

6.3. MAC address configuration

The following example shows how the operational state datastore could look like for an Ethernet interface without an explicit MAC address configured. The mac-address leaf always reports the actual operational MAC address that is in use. The bia-mac-address leaf always reports the default MAC address assigned to the hardware.

```
<?xml version="1.0" encoding="utf-8"?>
  <interfaces
    xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
    xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
      <phys-address>00:00:5E:00:53:30</phys-address>
      <ethernet-like
        xmlns="urn:ietf:params:xml:ns:yang:ietf-if-ethernet-like">
        <mac-address>00:00:5E:00:53:30</mac-address>
        <bia-mac-address>00:00:5E:00:53:30</bia-mac-address>
      </ethernet-like>
    </interface>
  </interfaces>
```

The following example shows the intended configuration for interface eth0 with an explicit MAC address configured.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces
    xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
    xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
      <ethernet-like
        xmlns="urn:ietf:params:xml:ns:yang:ietf-if-ethernet-like">
        <mac-address>00:00:5E:00:53:35</mac-address>
      </ethernet-like>
    </interface>
  </interfaces>
</config>
```

After the MAC address configuration has been successfully applied, the operational state datastore reporting the interface MAC address properties would contain the following, with the mac-address leaf updated to match the configured value, but the bia-mac-address leaf retaining the same value - which should never change.

```
<?xml version="1.0" encoding="utf-8"?>
<interfaces
 xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
 xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
  <interface>
    <name>eth0</name>
    <type>ianaift:ethernetCsmacd</type>
      <phys-address>00:00:5E:00:53:35</phys-address>
    <ethernet-like
      xmlns="urn:ietf:params:xml:ns:yang:ietf-if-ethernet-like">
      <mac-address>00:00:5E:00:53:35</mac-address>
      <bia-mac-address>00:00:5E:00:53:30</bia-mac-address>
    </ethernet-like>
  </interface>
</interfaces>
```

7. Acknowledgements

The authors wish to thank Eric Gray, Ing-Wher Chen, Jon Culver, Juergen Schoenwaelder, Ladislav Lhotka, Lou Berger, Mahesh Jethanandani, Martin Bjorklund, Michael Zitao, Neil Ketley, Qin Wu, William Lupton, Xufeng Liu, Andy Bierman, and Vladimir Vassilev for their helpful comments contributing to this document.

8. ChangeLog

XXX, RFC Editor, please delete this change log before publication.

8.1. Version -08

- Initial updates after WG LC comments.

8.2. Version -07

- Minor editorial updates

8.3. Version -06

- Remove reservable-bandwidth, based on Acee's suggestion
- Add examples
- Add additional state parameters for carrier-delay and dampening

8.4. Version -05

- Incorporate feedback from Andy Bierman

8.5. Version -04

- Incorporate feedback from Lada, some comments left as open issues.

8.6. Version -03

- Fixed incorrect module name references, and updated tree output

8.7. Version -02

- Minor changes only: Fix errors in when statements, use derived-from-or-self() for future proofing.

9. IANA Considerations

This document defines several new YANG module and the authors politely request that IANA assigns unique names to the two YANG module files contained within this document, and also appropriate URIs in the "IETF XML Registry".

10. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol RFC 6241 [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory to implement secure transport is SSH RFC 6242 [RFC6242]. The NETCONF access control model RFC 6536 [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of data nodes defined in this YANG module which are writable/creatable/deletable (i.e. config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g. edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

10.1. ietf-if-extensions.yang

The ietf-if-extensions YANG module contains various configuration leaves that affect the behavior of interfaces. Modifying these leaves can cause an interface to go down, or become unreliable, or to drop traffic forwarded over it. More specific details of the possible failure modes are given below.

The following leaf could cause the interface to go down and stop processing any ingress or egress traffic on the interface. It could also cause broadcast traffic storms.

- /if:interfaces/if:interface/loopback

The following leaves could cause instabilities at the interface link layer, and cause unwanted higher layer routing path changes if the leaves are modified, although they would generally only affect a device that had some underlying link stability issues:

- /if:interfaces/if:interface/carrier-delay/down

- /if:interfaces/if:interface/carrier-delay/up

- /if:interfaces/if:interface/dampening/half-life
- /if:interfaces/if:interface/dampening/reuse
- /if:interfaces/if:interface/dampening/suppress
- /if:interfaces/if:interface/dampening/max-suppress-time

The following leaves could cause traffic loss on the interface because the received or transmitted frames do not comply with the frame matching criteria on the interface and hence would be dropped:

- /if:interfaces/if:interface/encapsulation
- /if:interfaces/if:interface/max-frame-size
- /if:interfaces/if:interface/forwarding-mode

Changing the parent-interface leaf could cause all traffic on the affected interface to be dropped. The affected leaf is:

- /if:interfaces/if:interface/parent-interface

10.2. ietf-if-ethernet-like.yang

Generally, the configuration nodes in the ietf-if-ethernet-like YANG module are concerned with configuration that is common across all types of Ethernet-like interfaces. The module currently only contains a node for configuring the operational MAC address to use on an interface. Adding/modifying/deleting this leaf has the potential risk of causing protocol instability, excessive protocol traffic, and general traffic loss, particularly if the configuration change caused a duplicate MAC address to be present on the local network . The following leaf is affected:

- interfaces/interface/ethernet-like/mac-address

11. References

11.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

11.2. Informative References

[I-D.ietf-netmod-sub-intf-vlan-model] Wilton, R., Ball, D., tapsingh@cisco.com, t., and S. Sivaraj, "Sub-interface VLAN YANG Data Models", draft-ietf-netmod-sub-intf-vlan-model-05 (work in progress), March 2019.

[IEEE802.3.2] IEEE WG802.3 - Ethernet Working Group, "IEEE 802.3.2-2019", 2019.

[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, <https://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, <https://www.rfc-editor.org/info/rfc6242>.

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

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

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

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

Authors' Addresses

Robert Wilton (editor)
Cisco Systems

Email: rwilton@cisco.com

David Ball
Cisco Systems

Email: daviball@cisco.com

Tapraj Singh
Cisco Systems

Email: tapsingh@cisco.com

Selvakumar Sivaraj
Juniper Networks

Email: ssivaraj@juniper.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2020

A. Clemm
Y. Qu
Futurewei
J. Tantsura
Apstra
A. Bierman
YumaWorks

November 4, 2019

Comparison of NMDA datastores
draft-ietf-netmod-nmda-diff-03

Abstract

This document defines an RPC operation to compare management datastores that comply with the NMDA architecture.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://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 7, 2020.

Copyright Notice

Copyright (c) 2019 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 (https://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. Key Words 3
3. Definitions and Acronyms 3
4. Data Model Overview 4
5. YANG Data Model 6
6. Example 9
7. Performance Considerations 12
8. Possible Future Extensions 13
9. IANA Considerations 13
9.1. Updates to the IETF XML Registry 13
9.2. Updates to the YANG Module Names Registry 14
10. Security Considerations 14
11. Acknowledgments 15
12. References 15
12.1. Normative References 15
12.2. Informative References 16
Authors' Addresses 16

1. Introduction

The revised Network Management Datastore Architecture (NMDA) [RFC8342] introduces a set of new datastores that each hold YANG-defined data [RFC7950] and represent a different "viewpoint" on the data that is maintained by a server. New YANG datastores that are introduced include <intended>, which contains validated configuration data that a client application intends to be in effect, and <operational>, which contains at least conceptually operational state data (such as statistics) as well as configuration data that is actually in effect.

NMDA introduces in effect a concept of "lifecycle" for management data, allowing to clearly distinguish between data that is part of a configuration that was supplied by a user, configuration data that has actually been successfully applied and that is part of the operational state, and overall operational state that includes both applied configuration data as well as status and statistics.

As a result, data from the same management model can be reflected in multiple datastores. Clients need to specify the target datastore to be specific about which viewpoint of the data they want to access. This way, an application can differentiate whether they are (for example) interested in the configuration that has been applied and is

actually in effect, or in the configuration that was supplied by a client and that is supposed to be in effect.

Due to the fact that data can propagate from one datastore to another, it is possibly for differences between datastores to occur. Some of this is entirely expected, as there may be a time lag between when a configuration is given to the device and reflected in <intended>, until when it actually takes effect and is reflected in <operational>. However, there may be cases when a configuration item that was to be applied may not actually take effect at all or needs an unusually long time to do so. This can be the case due to certain conditions not being met, resource dependencies not being resolved, or even implementation errors in corner conditions.

When configuration that is in effect is different from configuration that was applied, many issues can result. It becomes more difficult to operate the network properly due to limited visibility of actual status which makes it more difficult to analyze and understand what is going on in the network. Services may be negatively affected (for example, breaking a service instance resulting in service is not properly delivered to a customer) and network resources be misallocated.

Applications can potentially analyze any differences between two datastores by retrieving the contents from both datastores and comparing them. However, in many cases this will be at the same time costly and extremely wasteful.

This document introduces a YANG data model which defines RPCs, intended to be used in conjunction with NETCONF [RFC6241] or RESTCONF [RFC8040], that allow a client to request a server to compare two NMDA datastores and report any differences.

2. Key Words

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Definitions and Acronyms

NMDA: Network Management Datastore Architecture

RPC: Remote Procedure Call

4. Data Model Overview

At the core of the solution is a new management operation, <compare>, that allows to compare two datastores for the same data. The operation checks whether there are any differences in values or in data nodes that are contained in either datastore, and returns any differences as output. The output is returned in the format specified in YANG-Patch [RFC8072].

The YANG data model defines the <compare> operation as a new RPC. The operation takes the following input parameters:

- source: The source identifies the datastore that will serve as reference for the comparison, for example <intended>.

- target: The target identifies the datastore to compare against the source.

- filter-spec: This is a choice between different filter constructs to identify the portions of the datastore to be retrieved. It acts as a node selector that specifies which data nodes are within the scope of the comparison and which nodes are outside the scope. This allows a comparison operation to be applied only to a specific portion of the datastore that is of interest, such as a particular subtree. (The filter dow not contain expressions that would match values data nodes, as this is not required by most use cases and would complicate the scheme, from implementation to dealing with race conditions.)

- all: When set, this parameter indicates that all differences should be included, including differences pertaining to schema nodes that exist in only one of the datastores. When this parameter is not included, a prefiltering step is automatically applied to exclude data from the comparison that does not pertain to both datastores: if the same schema node is not present in both datastores, then all instances of that schema node and all its descendants are excluded from the comparison. This allows client applications to focus on the differences that constitute true mismatches of instance data without needing to specify more complex filter constructs.

- exclude-origin: When set, this parameter indicates that origin metadata should not not be included as part of RPC output. When this parameter is omitted, origin metadata in comparisons that involve <operational> is by default included.

The operation provides the following output parameter:

- differences: This parameter contains the list of differences. Those differences are encoded per YANG-Patch data model defined in RFC8072. The YANG-Patch data model is augmented to indicate the value of source datastore nodes in addition to the patch itself that would need to be applied to the source to produce the target. When the target datastore is <operational>, "origin" metadata is included as part of the patch. Including origin metadata can help in some cases explain the cause of a difference, for example when a data node is part of <intended> but the origin of the same data node in <operational> is reported as "system".

The data model is defined in the ietf-nmda-compare YANG module. Its structure is shown in the following figure. The notation syntax follows [RFC8340].

```
module: ietf-nmda-compare
rpcs:
   +---x compare
      +---w input
      |  +---w source            identityref
      |  +---w target            identityref
      |  +---w all?              empty
      |  +---w exclude-origin?   empty
      |  +---w (filter-spec)?
      |     +--:(subtree-filter)
      |     |  +---w subtree-filter?
      |     +--:(xpath-filter)
      |        +---w xpath-filter?     yang:xpath1.0 {nc:xpath}?
      +--ro output
         +--ro (compare-response)?
            +--:(no-matches)
            |  +--ro no-matches?    empty
            +--:(differences)
               +--ro differences
                  +--ro yang-patch
                     +--ro patch-id    string
                     +--ro comment?    string
                     +--ro edit* [edit-id]
                        +--ro edit-id         string
                        +--ro operation       enumeration
                        +--ro target          target-resource-offset
                        +--ro point?          target-resource-offset
                        +--ro where?          enumeration
                        +--ro value?
                        +--ro source-value?
```

Structure of ietf-nmda-compare

5. YANG Data Model

```
<CODE BEGINS> file "ietf-nmda-compare@2019-11-04.yang"
module ietf-nmda-compare {

  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-nmda-compare";

  prefix cp;

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-datastores {
    prefix ds;
  }
  import ietf-yang-patch {
    prefix ypatch;
  }
  import ietf-netconf {
    prefix nc;
  }

  organization "IETF";
  contact
    "WG Web:   <http://tools.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>

     Author: Alexander Clemm
             <mailto:ludwig@clemm.org>

     Author: Yingzhen Qu
             <mailto:yqu@futurewei.com>

     Author: Jeff Tantsura
             <mailto:jefftant.ietf@gmail.com>

     Author: Andy Bierman
             <mailto:andy@yumaworks.com>";

  description
    "The YANG data model defines a new operation, <compare>, that
     can be used to compare NMDA datastores.

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
     NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
     'MAY', and 'OPTIONAL' in this document are to be interpreted as
     described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
```

```
        they appear in all capitals, as shown here.

        Copyright (c) 2019 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
        (https://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see the
        RFC itself for full legal notices.";

    revision 2019-11-04 {
      description
        "Initial revision";
      reference
        "RFC XXXX: Comparison of NMDA datastores";
    }

    /* RPC */
    rpc compare {
      description
        "NMDA compare operation.";
      input {
        leaf source {
          type identityref {
            base ds:datastore;
          }
          mandatory true;
          description
            "The source datastore to be compared.";
        }
        leaf target {
          type identityref {
            base ds:datastore;
          }
          mandatory true;
          description
            "The target datastore to be compared.";
        }
        leaf all {
          type empty;
          description
            "When this leaf is provided, all data nodes are compared,
             whether their schema node pertains to both datastores or
```

```
            not. When this leaf is omitted, a prefiltering step is
            automatically applied that excludes data nodes from the
            comparison that can occur in only one datastore but not
            the other.  Specifically, if one of the datastores
            (source or target) contains only configuration data and
            the other datastore is <operational>, data nodes for
            which config is false are excluded from the comparison.";
        }
        leaf exclude-origin {
          type empty;
          description
            "When this leaf is provided, origin metadata is not
             included as part of RPC output. When this leaf is
             omitted, origin metadata in comparisons that involve
             <operational> is by default included.";
        }
        choice filter-spec {
          description
            "Identifies the portions of the datastores to be
                 compared.";
          anydata subtree-filter {
            description
              "This parameter identifies the portions of the
               target datastore to retrieve.";
            reference "RFC 6241, Section 6.";
          }
          leaf xpath-filter {
            if-feature nc:xpath;
            type yang:xpath1.0;
            description
              "This parameter contains an XPath expression
               identifying the portions of the target
               datastore to retrieve.";
          }
        }
      }
      output {
        choice compare-response {
          description
            "Comparison results.";
          leaf no-matches {
            type empty;
            description
              "This leaf indicates that the filter did not match
               anything and nothing was compared.";
          }
          container differences {
            description
```

```
              "The list of differences, encoded per RFC8072 with an
               augmentation to include source values where
               applicable.";
            uses ypatch:yang-patch {
              augment "yang-patch/edit" {
                description
                  "Provide the value of the source of the patch,
                   respectively of the comparison, in addition to
                   the target value, where applicable.";
                anydata source-value {
                  when "../operation = 'delete'"
                    + "or ../operation = 'merge'"
                    + "or ../operation = 'move'"
                    + "or ../operation = 'replace'"
                    + "or ../operation = 'remove'";
                  description
                    "The anydata 'value' is only used for 'delete',
                     'move', 'merge', 'replace', and 'remove'
                     operations.";
                }
              }
            }
          }
        }
      }
    }
  }
  <CODE ENDS>
```

6. Example

The following example compares the difference between <operational> and <intended> for a subtree under "ospf". The subtree contains objects that are defined in a YANG data model for the management of OSPF defined in [I-D.ietf-ospf-yang]. The excerpt of the data model whose instantiation is basis of the comparison is as follows:

```
container ospf {
  leaf enable {
    type boolean;
  }
  leaf explicit-router-id {
    type rt-types:router-id;
  }
  leaf preference {
    type uint8;
  }
}
```

The contents of <intended> and <operational> datastores:

```
  <ospf xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
        or:origin="or:intended">
    <enable>true</enable>
    <explicit-router-id>2.2.2.2</explicit-router-id>
  </ospf>


  <ospf xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
        or:origin="or:operational">
    <enable>true</enable>
    <explicit-router-id>1.1.1.1</explicit-router-id>
    <preference>200</preference>
  </ospf>
```

<operational> contains one object that was not contained in <intended>, "preference". Another object, "explicit-router-id", has differences in values. A third object, "enable", is the same in both cases.

RPC request to compare <operational< (source of the comparison) with <intended>(target of the comparison):

```
     <rpc message-id="101"
         xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
       <compare xmlns="urn:ietf:params:xml:ns:yang:ietf-nmda-compare"
           xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
         <source>ds:operational</source>
         <target>ds:intended</target>
         <xpath-filter
           xmlns:rt="urn:ietf:params:xml:ns:yang:ietf-routing"
           xmlns:ospf="urn:ietf:params:xml:ns:yang:ietf-ospf">\
             /rt:routing/rt:control-plane-protocols\
             /rt:control-plane-protocol/ospf:ospf\
         </xpath-filter>
       </compare>
     </rpc>
```

RPC reply, when a difference is detected:

```
        <rpc-reply
             xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
             message-id="101">
          <differences
             xmlns="urn:ietf:params:xml:ns:yang:ietf-nmda-compare"
             xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin">
            <yang-patch>
              <patch-id>ospf router-id</patch-id>
              <comment>diff between operational and intended</comment>
              <edit>
                <edit-id>1</edit-id>
                <operation>replace</operation>
                <target>/ietf-ospf:explicit-router-id</target>
                <value>
                  <ospf:explicit-router-id
                    or:origin="or:system">1.1.1.1<explicit-router-id>
                </value>
                <source-value>
                   <ospf:explicit-router-id
                    or:origin="or:intended">2.2.2.2<explicit-router-id>
                </source-value>
                <edit-id>2</edit-id>
                <operation>create</operation>
                <target>/ietf-ospf:preference</target>
                <value>
                  <ospf:preference
                    or:origin="or:system">200<preference>
                </value>
              </edit>
            </yang-patch>
          </differences>
        </rpc-reply>
```

The same request in RESTCONF (using JSON format):

```
        POST /restconf/operations/ietf-nmda-compare:compare HTTP/1.1
        Host: example.com
        Content-Type: application/yang-data+json
        Accept: application/yang-data+json

        { "ietf-nmda-compare:input" {
           "source" : "ietf-datastores:operational",
           "target" : "ietf-datastores:intended".
           "xpath-filter" : \
               "/ietf-routing:routing/control-plane-protocols\
               /control-plane-protocol/ietf-ospf:ospf"
           }
        }
```

The same response in RESTCONF (using JSON format):

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2019 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+json

{ "ietf-nmda-compare:output" : {
    "differences" : {
      "ietf-yang-patch:yang-patch" : {
        "patch-id" : "ospf router-id",
        "comment" : "diff between operational and intended",
        "edit" : [
          {
            "edit-id" : "1",
            "operation" : "replace",
            "target" : "/ietf-ospf:explicit-router-id",
            "value" : {
               "ietf-ospf:explicit-router-id" : "1.1.1.1"
               "@ietf-ospf:explicit-router-id" : {
               "ietf-origin:origin" : "ietf-origin:system"
               }
            "source-value" : {
               "ietf-ospf:explicit-router-id" : "2.2.2.2"
               "@ietf-ospf:explicit-router-id" : {
               "ietf-origin:origin" : "ietf-origin:intended"
               }
            "edit-id" : "2",
            "operation" : "create",
            "target" : "/ietf-ospf:preference",
            "value" : {
               "ietf-ospf:preference" : "200"
               "@ietf-ospf:preference" : {
               "ietf-origin:origin" : "ietf-origin:system"
               }
          }
        ]
      }
    }
  }
}
```

7. Performance Considerations

The compare operation can be computationally expensive. While responsible client applications are expected to use the operation responsibly and sparingly only when warranted, implementations need

to be aware of the fact that excessive invocation of this operation will burden system resources and need to ensure that system performance will not be adversely impacted. One possibility for an implementation to mitigate against such a possibility is to limit the number of requests that is served to a client in any one time interval, rejecting requests made at a higher frequency than the implementation can reasonably sustain.

8. Possible Future Extensions

It is conceivable to extend the compare operation with a number of possible additional features in the future.

Specifically, it is possible to define an extension with an optional feature for dampening. This will allow clients to specify a minimum time period for which a difference must persist for it to be reported. This will enable clients to distinguish between differences that are only fleeting from ones that are not and that may represent a real operational issue and inconsistency within the device.

For this purpose, an additional input parameter can be added to specify the dampening period. Only differences that pertain for at least the dampening time are reported. A value of 0 or omission of the parameter indicates no dampening. Reporting of differences MAY correspondingly be delayed by the dampening period from the time the request is received.

To implement this feature, a server implementation might run a comparison when the RPC is first invoked and temporarily store the result. Subsequently, it could wait until after the end of the dampening period to check whether the same differences are still observed. The differences that still persist are then returned.

9. IANA Considerations

9.1. Updates to the IETF XML Registry

This document registers one URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

URI: urn:ietf:params:xml:ns:yang:ietf-nmda-compare

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

9.2. Updates to the YANG Module Names Registry

This document registers a YANG module in the YANG Module Names registry [RFC7950]. Following the format in [RFC7950], the following registration is requested:

name: ietf-nmda-compare

namespace: urn:ietf:params:xml:ns:yang:ietf-nmda-compare

prefix: cp

reference: RFC XXXX

10. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

The RPC operation defined in this YANG module, "compare", may be considered sensitive or vulnerable in some network environments. It is thus important to control access to this operation. This is the sensitivity/vulnerability of RPC operation "compare":

Comparing datastores for differences requires a certain amount of processing resources at the server. An attacker could attempt to attack a server by making a high volume of comparison requests. Server implementations can guard against such scenarios in several ways. For one, they can implement the NETCONF access control model in order to require proper authorization for requests to be made. Second, server implementations can limit the number of requests that they serve to a client in any one time interval, rejecting requests made at a higher frequency than the implementation can reasonably sustain.

11. Acknowledgments

We thank Rob Wilton, Martin Bjorklund, Mahesh Jethanandani, Lou Berger, Kent Watsen, Phil Shafer, Ladislav Lhotka, and Tim Carey for valuable feedback and suggestions.

12. References

12.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[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, <https://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, <https://www.rfc-editor.org/info/rfc6242>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8072] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Patch Media Type", RFC 8072, DOI 10.17487/RFC8072, February 2017, <https://www.rfc-editor.org/info/rfc8072>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

12.2. Informative References

[I-D.ietf-ospf-yang] Yeung, D., Qu, Y., Zhang, Z., Chen, I., and A. Lindem, "YANG Data Model for OSPF Protocol", draft-ietf-ospf-yang-29 (work in progress), October 2019.

Authors' Addresses

Alexander Clemm
Futurewei
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: ludwig@clemm.org

Yingzhen Qu
Futurewei
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: yqu@futurewei.com

Jeff Tantsura
Apstra

Email: jefftant.ietf@gmail.com

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2020

R. Wilton, Ed.
D. Ball
T. Singh
Cisco Systems
S. Sivaraj
Juniper Networks

November 4, 2019

Sub-interface VLAN YANG Data Models
draft-ietf-netmod-sub-intf-vlan-model-06

Abstract

This document defines YANG modules to add support for classifying traffic received on interfaces as Ethernet/VLAN framed packets to sub-interfaces based on the fields available in the Ethernet/VLAN frame headers. These modules allow configuration of Layer 3 and Layer 2 sub-interfaces (e.g. attachment circuits) that can interoperate with IETF based forwarding protocols; such as IP and L3VPN services; or L2VPN services like VPWS, VPLS, and EVPN. The sub-interfaces also interoperate with VLAN tagged traffic orginating from an IEEE 802.1Q compliant bridge.

The model differs from an IEEE 802.1Q bridge model in that the configuration is interface/sub-interface based as opposed to being based on membership of an 802.1Q VLAN bridge.

The YANG data models in this document conforms to the Network Management Datastore Architecture (NMDA) defined in RFC 8342.

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 https://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 7, 2020.

Copyright Notice

Copyright (c) 2019 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 (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
1.1. Terminology 3
1.2. Tree Diagrams 4
2. Objectives 4
2.1. Interoperability with IEEE 802.1Q compliant bridges . . . 4
3. L3 Interface VLAN Model 4
4. Flexible Encapsulation Model 5
5. L3 Interface VLAN YANG Module 7
6. Flexible Encapsulation YANG Module 10
7. Examples 19
7.1. Layer 3 sub-interfaces with IPv6 19
7.2. Layer 2 sub-interfaces with L2VPN 21
8. Acknowledgements 23
9. ChangeLog 23
9.1. WG version -05 24
9.2. WG version -04 24
9.3. WG version -03 24
9.4. WG version -02 24
9.5. WG version -01 24
9.6. Version -04 24
9.7. Version -03 24
10. IANA Considerations 25
11. Security Considerations 25
11.1. if-l3-vlan.yang 25
11.2. flexible-encapsulation.yang 26
12. References 27
12.1. Normative References 27
12.2. Informative References 28
Appendix A. Comparison with the IEEE 802.1Q Configuration Model 29
A.1. Sub-interface based configuration model overview 29
A.2. IEEE 802.1Q Bridge Configuration Model Overview 30

A.3. Possible Overlap Between the Two Models 31
Authors' Addresses 31

1. Introduction

This document defines two YANG [RFC7950] modules that augment the encapsulation choice YANG element defined in Interface Extensions YANG [I-D.ietf-netmod-intf-ext-yang] and the generic interfaces data model defined in [RFC8343]. The two modules provide configuration nodes to support classification of Ethernet/VLAN traffic to sub-interfaces, that can have interface based feature and service configuration applied to them.

The purpose of these models is to allow IETF defined forwarding protocols, such as IPv6 [RFC2460], Ethernet Pseudo Wires [RFC4448] and VPLS [RFC4761] [RFC4762] to be configurable via YANG when interoperating with VLAN tagged traffic received from an IEEE 802.1Q compliant bridge.

In the case of layer 2 Ethernet services, the flexible encapsulation module also supports flexible rewriting of the VLAN tags contained the in frame header.

For reference, a comparison between the sub-interface based YANG model documented in this draft and an IEEE 802.1Q bridge model is described in Appendix A.

In summary, the YANG modules defined in this internet draft are:

if-l3-vlan.yang - Defines the model for basic classification of VLAN tagged traffic to L3 transport services

flexible-encapsulation.yang - Defines the model for flexible classification of Ethernet/VLAN traffic to L2 transport services

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 RFC 2119 [RFC2119] RFC 8174 [RFC8174] when, and only when, they appear in all capitals, as shown here.

Sub-interface: A sub-interface is a small augmentation of a regular interface in the standard YANG module for Interface Management that represents a subset of the traffic handled by its parent interface. As such, it supports both configuration and operational data using any other YANG models that augment or reference interfaces in the

normal way. It is defined in Interface Extensions YANG [I-D.ietf-netmod-intf-ext-yang].

1.2. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

2. Objectives

The primary aim of the YANG modules contained in this draft is to provide the core model that is required to implement VLAN transport services on router based devices that is fully compatible with IEEE 802.1Q compliant bridges.

A secondary aim is for the modules to be structured in such a way that they can be cleanly extended in future.

2.1. Interoperability with IEEE 802.1Q compliant bridges

The modules defined in this document are designed to fully interoperate with IEEE 802.1Q compliant bridges. In particular, the models are restricted to only matching, adding, or rewriting the 802.1Q VLAN tags in frames in ways that are compatible with IEEE 802.1Q compliant bridges.

3. L3 Interface VLAN Model

The L3 Interface VLAN model provides appropriate leaves for termination of an 802.1Q VLAN tagged segment to a sub-interface based L3 service. It allows for termination of traffic with up to two 802.1Q VLAN tags.

The "if-l3-vlan" YANG module has the following structure:

```
module: ietf-if-l3-vlan
  augment /if:interfaces/if:interface/if-ext:encapsulation
            /if-ext:encaps-type:
    +--:(dot1q-vlan)
       +--rw dot1q-vlan
          +--rw outer-tag
          |  +--rw tag-type    dot1q-tag-type
          |  +--rw vlan-id     vlanid
          +--rw second-tag!
             +--rw tag-type    dot1q-tag-type
             +--rw vlan-id     vlanid
```

4. Flexible Encapsulation Model

The Flexible Encapsulation model is designed to allow for the flexible provisioning of layer 2 services. It provides the capability to classify Ethernet/VLAN frames received on an Ethernet trunk interface to sub-interfaces based on the fields available in the layer 2 headers. Once classified to sub-interfaces, it provides the capability to selectively modify fields within the layer 2 headers before the frame is handed off to the appropriate forwarding code for further handling.

The model supports a common core set of layer 2 header matches based on the 802.1Q tag type and VLAN Ids contained within the header up to a tag stack depth of two tags.

The model supports flexible rewrites of the layer 2 frame header for data frames as they are processed on the interface. It defines a set of standard tag manipulations that allow for the insertion, removal, or rewrite of one or two 802.1Q VLAN tags. The expectation is that manipulations are generally implemented in a symmetrical fashion, i.e. if a manipulation is performed on ingress traffic on an interface then the reverse manipulation is always performed on egress traffic out of the same interface. However, the model also allows for asymmetrical rewrites, which may be required to implement some forwarding models (such as E-Tree).

The final aim for the model design is for it to be cleanly extensible to add in additional match and rewrite criteria of the layer 2 header, such as matching on the source or destination MAC address, PCP or DEI fields in the 802.1Q tags, or the EtherType of the frame payload. Rewrites can also be extended to allow for modification of other fields within the layer 2 frame header.

The "flexible-encapsulation" YANG module has the following structure:

```
module: ietf-flexible-encapsulation
  augment /if:interfaces/if:interface/if-ext:encapsulation
            /if-ext:encaps-type:
    +--:(flexible)
       +--rw flexible
          +--rw match
          |  +--rw (match-type)
          |     +--:(default)
          |     |  +--rw default?                empty
          |     +--:(untagged)
          |     |  +--rw untagged?               empty
          |     +--:(dot1q-priority-tagged)
```

```
|      |  +--rw dot1q-priority-tagged
|      |     +--rw tag-type    dot1q-types:dot1q-tag-type
|      +--:(dot1q-vlan-tagged)
|         +--rw dot1q-vlan-tagged
|            +--rw outer-tag
|            |  +--rw tag-type    dot1q-tag-type
|            |  +--rw vlan-id     union
|            +--rw second-tag!
|            |  +--rw tag-type    dot1q-tag-type
|            |  +--rw vlan-id     union
|            +--rw match-exact-tags?   empty
|      +--rw rewrite {flexible-rewrites}?
|      |  +--rw (direction)?
|      |     +--:(symmetrical)
|      |     |  +--rw symmetrical
|      |     |     +--rw dot1q-tag-rewrite {dot1q-tag-rewrites}?
|      |     |        +--rw pop-tags?    uint8
|      |     |        +--rw push-tags!
|      |     |           +--rw outer-tag
|      |     |           |  +--rw tag-type    dot1q-tag-type
|      |     |           |  +--rw vlan-id     vlanid
|      |     |           +--rw second-tag!
|      |     |              +--rw tag-type    dot1q-tag-type
|      |     |              +--rw vlan-id     vlanid
|      |     +--:(asymmetrical) {asymmetric-rewrites}?
|      |        +--rw ingress
|      |        |  +--rw dot1q-tag-rewrite {dot1q-tag-rewrites}?
|      |        |     +--rw pop-tags?    uint8
|      |        |     +--rw push-tags!
|      |        |        +--rw outer-tag
|      |        |        |  +--rw tag-type    dot1q-tag-type
|      |        |        |  +--rw vlan-id     vlanid
|      |        |        +--rw second-tag!
|      |        |           +--rw tag-type    dot1q-tag-type
|      |        |           +--rw vlan-id     vlanid
|      |        +--rw egress
|      |           +--rw dot1q-tag-rewrite {dot1q-tag-rewrites}?
|      |              +--rw pop-tags?    uint8
|      |              +--rw push-tags!
|      |                 +--rw outer-tag
|      |                 |  +--rw tag-type    dot1q-tag-type
|      |                 |  +--rw vlan-id     vlanid
|      |                 +--rw second-tag!
|      |                    +--rw tag-type    dot1q-tag-type
|      |                    +--rw vlan-id     vlanid
|      +--rw local-traffic-default-encaps!
|         +--rw outer-tag
|         |  +--rw tag-type    dot1q-tag-type
```

```
               |  +--rw vlan-id     vlanid
               +--rw second-tag!
                  +--rw tag-type    dot1q-tag-type
                  +--rw vlan-id     vlanid
```

5. L3 Interface VLAN YANG Module

This YANG module augments the encapsultion container defined in Interface Extensions YANG [I-D.ietf-netmod-intf-ext-yang].

```
<CODE BEGINS> file "ietf-if-l3-vlan@2019-11-04.yang"
module ietf-if-l3-vlan {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-if-l3-vlan";

  prefix if-l3-vlan;

  import ietf-interfaces {
    prefix if;
  }

  import iana-if-type {
    prefix ianaift;
  }

  import ieee802-dot1q-types {
    prefix dot1q-types;
  }

  import ietf-if-extensions {
    prefix if-ext;
  }

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

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

     Editor:   Robert Wilton
               <mailto:rwilton@cisco.com>";

  description
    "This YANG module models L3 VLAN sub-interfaces
```

```
        Copyright (c) 2019 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
        (https://trustee.ietf.org/license-info).

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

    revision 2019-11-04 {
      description "Latest draft revision";

      reference
        "Internet-Draft draft-ietf-netmod-sub-intf-vlan-model-06";
    }

    /*
     * Add support for the 802.1Q VLAN encapsulation syntax on layer 3
     * terminated VLAN sub-interfaces.
     */
    augment "/if:interfaces/if:interface/if-ext:encapsulation/" +
            "if-ext:encaps-type" {
      when
          "derived-from-or-self(../if:type,
                                'ianaift:ethernetCsmacd') or
           derived-from-or-self(../if:type,
                                'ianaift:ieee8023adLag') or
           derived-from-or-self(../if:type,
                                'if-ext:ethSubInterface')" {
        description
          "Applies only to Ethernet-like interfaces and
           sub-interfaces";
      }

      description
        "Augment the generic interface encapsulation with an
         basic 802.1Q VLAN encapsulation for sub-interfaces.";

      /*
       * Matches a single VLAN Id, or a pair of VLAN Ids to classify
       * traffic into an L3 service.
       */
      case dot1q-vlan {
```

```
          container dot1q-vlan {
            must
              'count(../../if-ext:forwarding-mode) = 0 or ' +
              'derived-from-or-self(../../if-ext:forwarding-mode,' +
                                   '"if-ext:layer-3-forwarding")' {
                error-message
                  "If the interface forwarding-mode leaf is set then it
                   must be set to an identity that derives from
                   layer-3-forwarding";

                description
                  "The forwarding-mode leaf on an interface can
                   optionally be used to enforce consistency of
                   configuration";
              }


            description
              "Match VLAN tagged frames with specific VLAN Ids";
            container outer-tag {
              must
                'tag-type = "dot1q-types:s-vlan" or ' +
                'tag-type = "dot1q-types:c-vlan"' {

                error-message
                    "Only C-VLAN and S-VLAN tags can be matched";

                description
                "For IEEE 802.1Q interoperability, only C-VLAN and
                     S-VLAN tags can be matched";
              }

              description
                "Classifies traffic using the outermost VLAN tag on the
                 frame.";

              uses dot1q-types:dot1q-tag-classifier-grouping;
            }

            container second-tag {
              must
                '../outer-tag/tag-type = "dot1q-types:s-vlan" and ' +
                'tag-type = "dot1q-types:c-vlan"' {

                error-message
                  "When matching two tags, the outermost tag must be
                   specified and of S-VLAN type and the second outermost
                   tag must be of C-VLAN tag type";
```

```
              description
                "For IEEE 802.1Q interoperability, when matching two
                 tags, it is required that the outermost tag exists and
                 is an S-VLAN, and the second outermost tag is a
                 C-VLAN";
            }

            presence "Also classify on the second outermost VLAN tag";

            description
              "Classifies traffic using the second outermost VLAN tag
               on the frame.";

            uses dot1q-types:dot1q-tag-classifier-grouping;
          }
        }
      }
    }
  }
  <CODE ENDS>
```

6. Flexible Encapsulation YANG Module

This YANG module augments the encapsultion container defined in Interface Extensions YANG [I-D.ietf-netmod-intf-ext-yang].

This YANG module also augments the interface container defined in [RFC8343].

```
<CODE BEGINS> file "ietf-flexible-encapsulation@2019-11-04.yang"
module ietf-flexible-encapsulation {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-flexible-encapsulation";

  prefix flex;

  import ietf-interfaces {
    prefix if;
  }

  import iana-if-type {
    prefix ianaift;
  }

  import ietf-if-extensions {
```

```
      prefix if-ext;
    }

    import ieee802-dot1q-types {
      prefix dot1q-types;
    }

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

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

       Editor:   Robert Wilton
                 <mailto:rwilton@cisco.com>";

    description
      "This YANG module describes interface configuration for flexible
       VLAN matches and rewrites.

       Copyright (c) 2019 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
       (https://trustee.ietf.org/license-info).

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

    revision 2019-11-04 {
      description "Latest draft revision";

      reference
        "Internet-Draft draft-ietf-netmod-sub-intf-vlan-model-06";
    }

    feature flexible-rewrites {
      description
        "This feature indicates whether the network element supports
          specifying flexible rewrite operations";
    }
```

```
    feature asymmetric-rewrites {
      description
        "This feature indicates whether the network element supports
         specifying different rewrite operations for the ingress
         rewrite operation and egress rewrite operation.";
    }

    feature dot1q-tag-rewrites {
      description
        "This feature indicates whether the network element supports
         the flexible rewrite functionality specifying flexible 802.1Q
         tag rewrites";
    }

    /*
     * flexible-match grouping.
     *
     * This grouping represents a flexible match.
     *
     * The rules for a flexible match are:
     *     1. default, untagged, priority tag, or a stack of tags.
     *   - Each tag in the stack of tags matches:
     *      1. tag type (802.1Q or 802.1ad) +
     *      2. tag value:
     *        i. single tag
     *        ii. set of tag ranges/values.
     *        iii. "any" keyword
     */
    grouping flexible-match {
      description "Flexible match";
      choice match-type {
        mandatory true;
        description "Provides a choice of how the frames may be
                     matched";

        case default {
          description "Default match";
          leaf default {
            type empty;
            description
              "Default match.  Matches all traffic not matched to any
               other peer sub-interface by a more specific
               encapsulation.";
          } // leaf default
        } // case default

        case untagged {
          description "Match untagged Ethernet frames only";
```

```
          leaf untagged {
            type empty;
            description
              "Untagged match.  Matches all untagged traffic.";
          } // leaf untagged
        } // case untagged

        case dot1q-priority-tagged {
          description
            "Match 802.1Q priority tagged Ethernet frames only";

          container dot1q-priority-tagged {
            description "802.1Q priority tag match";
            leaf tag-type {
              type dot1q-types:dot1q-tag-type;
              mandatory true;
              description "The 802.1Q tag type of matched priority
                           tagged packets";
            }
          }
        }

        case dot1q-vlan-tagged {
          container dot1q-vlan-tagged {
            description "Matches VLAN tagged frames";

            container outer-tag {
              must
                'tag-type = "dot1q-types:s-vlan" or ' +
                'tag-type = "dot1q-types:c-vlan"' {

                error-message
                  "Only C-VLAN and S-VLAN tags can be matched";

                description
                  "For IEEE 802.1Q interoperability, only C-VLAN and
                   S-VLAN tags can be matched";
              }

              description
                "Classifies traffic using the outermost VLAN tag on the
                 frame.";

              uses
                'dot1q-types:'+
                'dot1q-tag-ranges-or-any-classifier-grouping';
            }
```

```
            container second-tag {
              must
                '../outer-tag/tag-type = "dot1q-types:s-vlan" and ' +
                'tag-type = "dot1q-types:c-vlan"' {

                error-message
                  "When matching two tags, the outermost tag must be
                   specified and of S-VLAN type and the second
                   outermost tag must be of C-VLAN tag type";

                description
                  "For IEEE 802.1Q interoperability, when matching two
                   tags, it is required that the outermost tag exists
                   and is an S-VLAN, and the second outermost tag is a
                   C-VLAN";
              }

              presence "Also classify on the second VLAN tag";

              description
                "Classifies traffic using the second outermost VLAN tag
                 on the frame.";

              uses
                'dot1q-types:'+
                'dot1q-tag-ranges-or-any-classifier-grouping';
            }

            leaf match-exact-tags {
              type empty;
              description
                "If set, indicates that all 802.1Q VLAN tags in the
                 Ethernet frame header must be explicitly matched, i.e.
                 the EtherType following the matched tags must not be a
                 802.1Q tag EtherType.  If unset then extra 802.1Q VLAN
                 tags are allowed.";
            }
          }
        }
      } // encaps-type
    }

    /*
     * Grouping for tag-rewrite that can be expressed either
     * symmetrically, or in the ingress and/or egress directions
     * independently.
     */
    grouping dot1q-tag-rewrite {
```

```
      description "Flexible rewrite";
      leaf pop-tags {
        type uint8 {
          range 1..2;
        }
        description "The number of tags to pop (or translate if used in
                     conjunction with push-tags)";
      }

      container push-tags {
        presence
          "802.1Q tags are pushed or translated";
        description "The 802.1Q tags to push (or translate if used in
                     conjunction with pop-tags)";

        container outer-tag {
          must
            'tag-type = "dot1q-types:s-vlan" or ' +
            'tag-type = "dot1q-types:c-vlan"' {

            error-message
              "Only C-VLAN and S-VLAN tags can be pushed";

            description
              "For IEEE 802.1Q interoperability, only C-VLAN and S-VLAN
               tags can be pushed";
          }

          description
            "The outermost VLAN tag to push onto the frame.";
          uses dot1q-types:dot1q-tag-classifier-grouping;
        }

        container second-tag {
          must
            '../outer-tag/tag-type = "dot1q-types:s-vlan" and ' +
            'tag-type = "dot1q-types:c-vlan"' {

            error-message
              "When pushing/rewriting two tags, the outermost tag must
               be specified and of S-VLAN type and the second outermost
               tag must be of C-VLAN tag type";

            description
              "For IEEE 802.1Q interoperability, when pushing two tags,
               it is required that the outermost tag exists and is an
               S-VLAN, and the second outermost tag is a C-VLAN";
          }
```

```
          presence
            "In addition to the first tag, also push/rewrite a second
             VLAN tag.";

          description
            "The second outermost VLAN tag to push onto the frame.";

          uses dot1q-types:dot1q-tag-classifier-grouping;
        }
      }
    }

    /*
     * Grouping for all flexible rewrites of fields in the L2 header.
     *
     * This currently only includes flexible tag rewrites, but is
     * designed to be extensible to cover rewrites of other fields in
     * the L2 header if required.
     */
    grouping flexible-rewrite {
      description "Flexible rewrite";

      /*
       * Tag rewrite.
       *
       * All tag rewrites are formed using a combination of pop-tags
       * and push-tags operations.
       */
      container dot1q-tag-rewrite {
        if-feature dot1q-tag-rewrites;
        description "Tag rewrite.  Translate operations are expressed
                     as a combination of tag push and pop operations.";
        uses dot1q-tag-rewrite;
      }
    }
    augment "/if:interfaces/if:interface/if-ext:encapsulation/" +
            "if-ext:encaps-type" {
      when
          "derived-from-or-self(../if:type,
                                'ianaift:ethernetCsmacd') or
           derived-from-or-self(../if:type,
                                'ianaift:ieee8023adLag') or
           derived-from-or-self(../if:type,
                                'if-ext:ethSubInterface')" {
        description
          "Applies only to Ethernet-like interfaces and
           sub-interfaces";
      }
```

```
      description
        "Add flexible match and rewrite for VLAN sub-interfaces";

      /*
       * A flexible encapsulation allows for the matching of ranges and
       * sets of VLAN Ids.  The structure is also designed to be
       * extended to allow for matching/rewriting other fields within
       * the L2 frame header if required.
       */
      case flexible {
        description "Flexible encapsulation and rewrite";
        container flexible {
          description "Flexible encapsulation and rewrite";

          container match {
            description
              "The match used to classify frames to this interface";
            uses flexible-match;
          }

          container rewrite {
            if-feature flexible-rewrites;
            description "L2 frame rewrite operations";
            choice direction {
              description
                "Whether the rewrite policy is symmetrical or
                 asymmetrical";
              case symmetrical {
                container symmetrical {
                  uses flexible-rewrite;
                  description
                    "Symmetrical rewrite.  Expressed in the ingress
                     direction, but the reverse operation is applied to
                     egress traffic";
                }
              }

              /*
               * Allow asymmetrical rewrites to be specified.
               */
              case asymmetrical {
                if-feature asymmetric-rewrites;
                description "Asymmetrical rewrite";
                container ingress {
                  uses flexible-rewrite;
                  description "Ingress rewrite";
                }
                container egress {
```

```
            uses flexible-rewrite;
            description "Egress rewrite";
          }
        }
      }
    }

    /*
     * For encapsulations that match a range of VLANs (or Any),
     * allow configuration to specify the default 802.1Q VLAN tag
     * values to use for any traffic that is locally sourced from
     * an interface on the device.
     */
    container local-traffic-default-encaps {
      presence
        "A local traffic default encapsulation has been
         specified";
      description
        "The 802.1Q VLAN tags to use by default for locally
         sourced traffic";

      container outer-tag {
        must
          'tag-type = "dot1q-types:s-vlan" or ' +
          'tag-type = "dot1q-types:c-vlan"' {

          error-message
            "Only C-VLAN and S-VLAN tags can be matched";

          description
            "For IEEE 802.1Q interoperability, only C-VLAN and
             S-VLAN tags can be matched";
        }

        description
          "The outermost VLAN tag for locally sourced traffic";

        uses dot1q-types:dot1q-tag-classifier-grouping;
      }

      container second-tag {
        must
          '../outer-tag/tag-type = "dot1q-types:s-vlan" and ' +
           'tag-type = "dot1q-types:c-vlan"' {

        error-message
          "When specifying two tags, the outermost tag must be
           specified and of S-VLAN type and the second outermost
```

```
                   tag must be of C-VLAN tag type";

                description
                  "For IEEE 802.1Q interoperability, when specifying two
                   tags, it is required that the outermost tag exists and
                   is an S-VLAN, and the second outermost tag is a
                   C-VLAN";
                }

                presence
                  "Indicates existence of a second outermost VLAN tag.";

                description
                  "The second outermost VLAN tag for locally sourced
                   traffic";

                uses dot1q-types:dot1q-tag-classifier-grouping;
              }
            }
          }
        }
      }
    }
    <CODE ENDS>
```

7. Examples

The following sections give examples of configuring a sub-interface supporting L3 forwarding, and also a sub-interface being used in conjunction with the IETF L2VPN YANG model [I-D.ietf-bess-l2vpn-yang].

7.1. Layer 3 sub-interfaces with IPv6

This example illustrates a layer 3 sub-interface configured to match traffic with a S-VLAN tag of 10, and C-VLAN tag of 20.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces
  xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
  xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
  xmlns:dot1q-types="urn:ieee:std:802.1Q:yang:ieee802-dot1q-types"
  xmlns:if-cmn="urn:ietf:params:xml:ns:yang:ietf-interfaces-common">
    <interface>
      <name>eth0</name>
```

```
        <type>ianaift:ethernetCsmacd</type>
      </interface>
      <interface>
        <name>eth0.1</name>
        <type>ianaift:l2vlan</type>
        <if-cmn:parent-interface>eth0</if-cmn:parent-interface>
        <if-cmn:encapsulation>
          <dot1q-vlan
           xmlns="urn:ietf:params:xml:ns:yang:ietf-if-l3-vlan">
            <outer-tag>
              <tag-type>dot1q-types:s-vlan</tag-type>
              <vlan-id>10</vlan-id>
            </outer-tag>
            <second-tag>
              <tag-type>dot1q-types:c-vlan</tag-type>
              <vlan-id>20</vlan-id>
            </second-tag>
          </dot1q-vlan>
        </if-cmn:encapsulation>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <forwarding>true</forwarding>
          <address>
            <ip>2001:db8::10</ip>
            <prefix-length>32</prefix-length>
          </address>
        </ipv6>
      </interface>
      <interface>
        <name>eth0.2</name>
        <type>ianaift:l2vlan</type>
        <if-cmn:parent-interface>eth0</if-cmn:parent-interface>
        <if-cmn:encapsulation>
          <dot1q-vlan
           xmlns="urn:ietf:params:xml:ns:yang:ietf-if-l3-vlan">
            <outer-tag>
              <tag-type>dot1q-types:s-vlan</tag-type>
              <vlan-id>11</vlan-id>
            </outer-tag>
          </dot1q-vlan>
        </if-cmn:encapsulation>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <forwarding>true</forwarding>
          <address>
            <ip>2001:db8::11</ip>
            <prefix-length>32</prefix-length>
          </address>
        </ipv6>
      </interface>
```

```
    </interfaces>
  </config>
```

7.2. Layer 2 sub-interfaces with L2VPN

This example illustrates a layer 2 sub-interface 'eth0.3' configured to match traffic with a S-VLAN tag of 10, and C-VLAN tag of 21; and both tags removed before the traffic is passed off to the L2VPN service.

It also illustrates another sub-interface 'eth1.0' under a separate physical interface configured to match traffic with a C-VLAN of 50, and the tag removed before traffic is given to any service. Sub-interface 'eth1.0' is not currently bound to any service and hence traffic classifed to that sub-interface is dropped.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces
  xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
  xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
  xmlns:dot1q-types="urn:ieee:std:802.1Q:yang:ieee802-dot1q-types"
  xmlns:if-cmn="urn:ietf:params:xml:ns:yang:ietf-interfaces-common">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
    </interface>
    <interface>
      <name>eth0.3</name>
      <type>ianaift:l2vlan</type>
      <if-cmn:parent-interface>eth0</if-cmn:parent-interface>
      <if-cmn:encapsulation>
        <flexible
    xmlns="urn:ietf:params:xml:ns:yang:ietf-flexible-encapsulation">
          <match>
            <dot1q-vlan-tagged>
              <outer-tag>
                <tag-type>dot1q-types:s-vlan</tag-type>
                <vlan-id>10</vlan-id>
              </outer-tag>
              <second-tag>
                <tag-type>dot1q-types:c-vlan</tag-type>
                <vlan-id>21</vlan-id>
              </second-tag>
            </dot1q-vlan-tagged>
          </match>
```

```
            <rewrite>
              <symmetrical>
                <dot1q-tag-rewrite>
                  <pop-tags>2</pop-tags>
                </dot1q-tag-rewrite>
              </symmetrical>
            </rewrite>
          </flexible>
        </if-cmn:encapsulation>
      </interface>
      <interface>
        <name>eth1</name>
        <type>ianaift:ethernetCsmacd</type>
      </interface>
      <interface>
        <name>eth1.0</name>
        <type>ianaift:l2vlan</type>
        <if-cmn:parent-interface>eth0</if-cmn:parent-interface>
        <if-cmn:encapsulation>
          <flexible
      xmlns="urn:ietf:params:xml:ns:yang:ietf-flexible-encapsulation">
            <match>
              <dot1q-vlan-tagged>
                <outer-tag>
                  <tag-type>dot1q-types:c-vlan</tag-type>
                  <vlan-id>50</vlan-id>
                </outer-tag>
              </dot1q-vlan-tagged>
            </match>
            <rewrite>
              <symmetrical>
                <dot1q-tag-rewrite>
                  <pop-tags>1</pop-tags>
                </dot1q-tag-rewrite>
              </symmetrical>
            </rewrite>
          </flexible>
        </if-cmn:encapsulation>
      </interface>
    </interfaces>
    <network-instances
        xmlns="urn:ietf:params:xml:ns:yang:ietf-network-instance">
      <network-instance
       xmlns:l2vpn="urn:ietf:params:xml:ns:yang:ietf-l2vpn">
        <name>p2p-l2-1</name>
        <description>Point to point L2 service</description>
        <l2vpn:type>l2vpn:vpws-instance-type</l2vpn:type>
        <l2vpn:signaling-type>
```

```
          l2vpn:ldp-signaling
        </l2vpn:signaling-type>
        <endpoint xmlns="urn:ietf:params:xml:ns:yang:ietf-l2vpn">
          <name>local</name>
          <ac>
            <name>eth0.3</name>
          </ac>
        </endpoint>
        <endpoint xmlns="urn:ietf:params:xml:ns:yang:ietf-l2vpn">
          <name>remote</name>
          <pw>
            <name>pw1</name>
          </pw>
        </endpoint>
        <vsi-root>
        </vsi-root>
      </network-instance>
    </network-instances>
    <pseudowires
        xmlns="urn:ietf:params:xml:ns:yang:ietf-pseudowires">
      <pseudowire>
        <name>pw1</name>
        <configured-pw>
          <peer-ip>2001:db8::50></peer-ip>
          <pw-id>100</pw-id>
        </configured-pw>
      </pseudowire>
    </pseudowires>
  </config>
```

8. Acknowledgements

The authors would particularly like to thank John Messenger, Glenn Parsons, and Dan Romascanu for their help progressing this draft.

The authors would also like to thank Alex Campbell, Eric Gray, Giles Heron, Marc Holness, Iftekhar Hussain, Neil Ketley, William Lupton, John Messenger, Glenn Parsons, Ludwig Pauwels, Joseph White, Vladimir Vassilev, and members of the IEEE 802.1 WG for their helpful reviews and feedback on this draft.

9. ChangeLog

XXX, RFC Editor, please delete this change log before publication.

9.1. WG version -05

- Incorporate feedback from IEEE 802.1 WG, John Messenger in particular.
- Adding must contraints to ensure outer tags are always matched to C-VLAN and S-VLAN tags.
- Fixed bug where second tag could be matched without outer tag, and where tags must not be specified.

9.2. WG version -04

- Added examples

9.3. WG version -03

- Fix namespace bug in XPath identity references, removed extraneous 'dot1q-tag' containers.

9.4. WG version -02

- Use explicit containers for outer and inner tags rather than lists.

9.5. WG version -01

- Tweaked the abstract.
- Removed unnecessary feature for the L3 sub-interface module.
- Update the 802.1Qcp type references.
- Remove extra tag container for L3 sub-interfaces YANG.

9.6. Version -04

- IEEE 802.1 specific types have been removed from the draft. These are now referenced from the 802.1Qcp draft YANG modules.
- Fixed errors in the xpath expressions.

9.7. Version -03

- Incorporates feedback received from presenting to the IEEE 802.1 WG.

- Updates the modules for double tag matches/rewrites to restrict the outer tag type to S-VLAN and inner tag type to C-VLAN.

- Updates the introduction to indicate primary use case is for IETF forwarding protocols.

- Updates the objectives to make IEEE 802.1Q bridge interoperability a key objective.

10. IANA Considerations

This document defines two new YANG module and the authors politely request that IANA assigns unique names to the YANG module files contained within this draft, and also appropriate URIs in the "IETF XML Registry".

11. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol RFC 6241 [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory to implement secure transport is SSH RFC 6242 [RFC6242]. The NETCONF access control model RFC 6536 [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of data nodes defined in this YANG module which are writable/creatable/deletable (i.e. config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g. edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

11.1. if-l3-vlan.yang

The nodes in the if-l3-vlan YANG module are concerned with matching particular frames received on the network device to connect them to a layer 3 forwarding instance, and as such adding/modifying/deleting these nodes has a high risk of causing traffic to be lost because it is not being classified correctly, or is being classified to a separate sub-interface. The nodes, all under the subtree /interfaces/interface/encapsulation/dot1q-vlan, that are sensitive to this are:

- outer-tag/tag-type

- outer-tag/vlan-id

- second-tag/tag-type
- second-tag/vlan-id

11.2. flexible-encapsulation.yang

There are many nodes in the flexible-encapsulation YANG module that are concerned with matching particular frames received on the network device, and as such adding/modifying/deleting these nodes has a high risk of causing traffic to be lost because it is not being classified correctly, or is being classified to a separate sub-interface. The nodes, all under the subtree /interfaces/interface/encapsulation/flexible/match, that are sensitive to this are:

- default
- untagged
- dot1q-priority-tagged
- dot1q-priority-tagged/tag-type
- dot1q-vlan-tagged/outer-tag/vlan-type
- dot1q-vlan-tagged/outer-tag/vlan-id
- dot1q-vlan-tagged/second-tag/vlan-type
- dot1q-vlan-tagged/second-tag/vlan-id

There are also many modes in the flexible-encapsulation YANG module that are concerned with rewriting the fields in the L2 header for particular frames received on the network device, and as such adding/modifying/deleting these nodes has a high risk of causing traffic to be dropped or incorrectly processed on peer network devices, or it could cause layer 2 tunnels to go down due to a mismatch in negotiated MTU. The nodes, all under the subtree /interfaces/interface/encapsulation/flexible/rewrite, that are sensitive to this are:

- symmetrical/dot1q-tag-rewrite/pop-tags
- symmetrical/dot1q-tag-rewrite/push-tags/outer-tag/tag-type
- symmetrical/dot1q-tag-rewrite/push-tags/outer-tag/vlan-id
- symmetrical/dot1q-tag-rewrite/push-tags/second-tag/tag-type

- symmetrical/dot1q-tag-rewrite/push-tags/second-tag/vlan-id
- asymmetrical/ingress/dot1q-tag-rewrite/pop-tags
- asymmetrical/ingress/dot1q-tag-rewrite/push-tags/outer-tag/tag-type
- asymmetrical/ingress/dot1q-tag-rewrite/push-tags/outer-tag/vlan-id
- asymmetrical/ingress/dot1q-tag-rewrite/push-tags/second-tag/tag-type
- asymmetrical/ingress/dot1q-tag-rewrite/push-tags/second-tag/vlan-id
- asymmetrical/egress/dot1q-tag-rewrite/pop-tags
- asymmetrical/egress/dot1q-tag-rewrite/push-tags/outer-tag/tag-type
- asymmetrical/egress/dot1q-tag-rewrite/push-tags/outer-tag/vlan-id
- asymmetrical/egress/dot1q-tag-rewrite/push-tags/second-tag/tag-type
- asymmetrical/egress/dot1q-tag-rewrite/push-tags/second-tag/vlan-id

Nodes in the flexible-encapsulation YANG module that are concerned with the VLAN tags to use for traffic sourced from the network element could cause protocol sessions (such as CFM) to fail if they are added, modified or deleted. The nodes, all under the subtree /interfaces/interface/flexible-encapsulation/local-traffic-default-encaps that are sensitive to this are:

- outer-tag/vlan-type
- outer-tag/vlan-id
- second-tag/vlan-type
- second-tag/vlan-id

12. References

12.1. Normative References

[I-D.ietf-netmod-intf-ext-yang]
Wilton, R., Ball, D., tsingh@juniper.net, t., and S. Sivaraj, "Common Interface Extension YANG Data Models", draft-ietf-netmod-intf-ext-yang-07 (work in progress), March 2019.

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

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

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

12.2. Informative References

[dot1Qcp] Holness, M., "IEEE 802.1Qcp-2018 Bridges and Bridged Networks - Amendment: YANG Data Model", 2018.

[I-D.ietf-bess-l2vpn-yang]
Shah, H., Brissette, P., Chen, I., Hussain, I., Wen, B., and K. Tiruveedhula, "YANG Data Model for MPLS-based L2VPN", draft-ietf-bess-l2vpn-yang-10 (work in progress), July 2019.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <https://www.rfc-editor.org/info/rfc2460>.

[RFC4448] Martini, L., Ed., Rosen, E., El-Aawar, N., and G. Heron, "Encapsulation Methods for Transport of Ethernet over MPLS Networks", RFC 4448, DOI 10.17487/RFC4448, April 2006, <https://www.rfc-editor.org/info/rfc4448>.

[RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <https://www.rfc-editor.org/info/rfc4761>.

[RFC4762] Lasserre, M., Ed. and V. Kompella, Ed., "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling", RFC 4762, DOI 10.17487/RFC4762, January 2007, <https://www.rfc-editor.org/info/rfc4762>.

[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, <https://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, <https://www.rfc-editor.org/info/rfc6242>.

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

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

Appendix A. Comparison with the IEEE 802.1Q Configuration Model

In addition to the sub-interface based YANG model proposed here, the IEEE 802.1Q working group has developed a YANG model for the configuration of 802.1Q VLANs. This raises the valid question as to whether the models overlap and whether it is necessary or beneficial to have two different models for superficially similar constructs. This section aims to answer that question by summarizing and comparing the two models.

A.1. Sub-interface based configuration model overview

The key features of the sub-interface based configuration model can be summarized as:

- The model is primarily designed to enable layer 2 and layer 3 services on Ethernet interfaces that can be defined in a very flexible way to meet the varied requirements of service providers.

- Traffic is classified from an Ethernet-like interface to sub-interfaces based on fields in the layer 2 header. This is often based on VLAN Ids contained in the frame, but the model is extensible to other arbitrary fields in the frame header.

- Sub-interfaces are just a type of if:interface and hence support any feature configuration YANG models that can be applied generally to interfaces. For example, QoS or ACL models that reference if:interface can be applied to the sub-interfaces, or the sub-interface can be used as an Access Circuit in L2VPN or L3VPN models that reference if:interface.

- In the sub-interface based configuration model, the classification of traffic arriving on an interface to a given sub-interface, based on fields in the layer 2 header, is completely independent of how the traffic is forwarded. The sub-interface can be referenced (via references to if:interface) by other models that specify how traffic is forwarded; thus sub-interfaces can support multiple different forwarding paradigms, including but not limited to: layer 3 (IPv4/IPv6), layer 2 pseudowires (over MPLS or IP), VPLS instances, EVPN instance.

- The model is flexible in the scope of the VLAN Identifier space. I.e. by default VLAN Ids can be scoped locally to a single Ethernet-like trunk interface, but the scope is determined by the forwarding paradigm that is used.

A.2. IEEE 802.1Q Bridge Configuration Model Overview

The key features of the IEEE 802.1Q bridge configuration model can be summarized as:

- Each VLAN bridge component has a set of Ethernet interfaces that are members of that bridge. Sub-interfaces are not used, nor required in the 802.1Q bridge model.

- Within a VLAN bridge component, the VLAN tag in the packet is used, along with the destination MAC address, to determine how to forward the packet. Other forwarding paradigms are not supported by the 802.1Q model.

- Classification of traffic to a VLAN bridge component is based only on the Ethernet interface that it arrived on.

- VLAN Identifiers are scoped to a VLAN bridge component. Often devices only support a single bridge component and hence VLANs are scoped globally within the device.

- Feature configuration is specified in the context of the bridge, or particular VLANs on a bridge.

A.3. Possible Overlap Between the Two Models

Both models can be used for configuring similar basic layer 2 forwarding topologies. The 802.1Q bridge configuration model is optimised for configuring Virtual LANs that span across enterprises and data centers.

The sub-interface model can also be used for configuring equivalent Virtual LAN networks that span across enterprises and data centers, but often requires more configuration to be able to configure the equivalent constructs to the 802.1Q bridge model.

The sub-interface model really excels when implementing flexible L2 and L3 services, where those services may be handled on the same physical interface, and where the VLAN Identifier is being solely used to identify the customer or service that is being provided rather than a Virtual LAN. The sub-interface model provides more flexibility as to how traffic can be classified, how features can be applied to traffic streams, and how the traffic is to be forwarded.

Conversely, the 802.1Q bridge model can also be use to implement L2 services in some scenarios, but only if the forwarding paradigm being used to implement the service is the native Ethernet forwarding specified in 802.1Q - other forwarding paradigms such as pseudowires or VPLS are not supported. The 802.1Q bridge model does not implement L3 services at all, although this can be partly mitigated by using a virtual L3 interface construct that is a separate logical Ethernet-like interface which is a member of the bridge.

In conclusion, it is valid for both of these models to exist since they have different deployment scenarios for which they are optimized. Devices may choose which of the models (or both) to implement depending on what functionality the device is being designed for.

Authors' Addresses

Robert Wilton (editor)
Cisco Systems

Email: rwilton@cisco.com

David Ball
Cisco Systems

Email: daviball@cisco.com

Tapraj Singh
Cisco Systems

Email: tapsingh@cisco.com

Selvakumar Sivaraj
Juniper Networks

Email: ssivaraj@juniper.net

Netmod
Internet-Draft
Intended status: Standards Track
Expires: February 13, 2020

B. Lengyel
Ericsson
B. Claise
Cisco Systems, Inc.

August 12, 2019

YANG Instance Data File Format
draft-ietf-netmod-yang-instance-file-format-04

Abstract

There is a need to document data defined in YANG models when a live server is not available. Data is often needed already at design or implementation time or needed by groups that do not have a live running server available. This document specifies a standard file format for YANG instance data (which follows the syntax and semantic from existing YANG models, re-using the same format as the reply to a <get> operation/request) and decorates it with metadata.

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 https://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 February 13, 2020.

Copyright Notice

Copyright (c) 2019 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 (https://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. Terminology 2
2. Introduction 3
2.1. Principles 4
3. Instance Data File Format 4
3.1. Specifying the Content Schema 6
3.1.1. Inline Method 7
3.1.2. Simplified-Inline Method 7
3.1.3. URI Method 7
3.2. Examples 8
4. Data Life cycle 12
5. Delivery of Instance Data 13
6. Backwards Compatibility 13
7. Yang Instance Data Model 13
7.1. Tree Diagram 13
7.2. YANG Model 14
8. Security Considerations 18
9. IANA Considerations 18
9.1. URI Registration 19
9.2. YANG Module Name Registration 19
10. Acknowledgments 19
11. References 19
11.1. Normative References 19
11.2. Informative References 20
Appendix A. Open Issues 21
Appendix B. Changes between revisions 21
Appendix C. Detailed Use Cases - Non-Normative 22
C.1. Use Cases 22
C.1.1. Use Case 1: Early Documentation of Server Capabilities 22
C.1.2. Use Case 2: Preloading Data 24
C.1.3. Use Case 3: Documenting Factory Default Settings . . 24
Authors' Addresses 24

1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 RFC 2119 [RFC2119] RFC 8174 [RFC8174] when, and only when, they appear in all capitals, as shown here.

Instance Data Set: A named set of data items decorated with metadata that can be used as instance data in a YANG data tree.

Instance Data File: A file containing an instance data set formatted according to the rules described in this document.

Content-schema: A set of YANG modules with their revision,suupported features and deviations for which the instance data set contains instance data

Content defining Yang module(s): YANG module(s) that make up the content-schema

YANG Instance Data, or just instance data for short, is data that could be stored in a datastore and whose syntax and semantics is defined by YANG models.

The term Server is used as defined in [RFC8342]

2. Introduction

There is a need to document data defined in YANG models when a live server is not available. Data is often needed already at design or implementation time or needed by groups that do not have a live running server available. To facilitate this off-line delivery of data this document specifies a standard format for YANG instance data sets and YANG instance data files.

The following is a list of already implemented and potential use cases.

UC1 Documentation of server capabilities

UC2 Preloading default configuration data

UC3 Documenting Factory Default Settings

UC4 Instance data used as backup

UC5 Storing the configuration of a device, e.g. for archive or audit purposes

UC6 Storing diagnostics data

UC7 Allowing YANG instance data to potentially be carried within other IPC message formats

UC8 Default instance data used as part of a templating solution

UC9 Providing data examples in RFCs or internet drafts

In Appendix C we describe the first three use cases in detail.

There are many and varied use cases where YANG instance data could be used. We do not want to limit future uses of instance data sets, so specifying how and when to use Yang instance data is out of scope for this document. It is anticipated that other documents will define specific use cases. Use cases are listed here only to indicate the need for this work.

2.1. Principles

The following is a list of the basic principles of the instance data format:

P1 Two standard formats are based on the XML and the JSON encoding

P2 Re-use existing formats similar to the response to a <get> operation/request

P3 Add metadata about the instance data set (Section 3, Paragraph 9)

P4 A YANG instance data set may contain data for many YANG modules

P5 Instance data may include configuration data, state data or a mix of the two

P6 Partial data sets are allowed

P7 YANG instance data format may be used for any data for which YANG module(s) are defined and available to the reader, independent of whether the module is actually implemented by a server

3. Instance Data File Format

A YANG instance data file MUST contain a single instance data set and no additional data.

The format of the instance data set is defined by the ietf-yang-instance-data YANG module. It is made up of a header part and content-data. The header part carries metadata for the instance data set. The content-data, defined as an anydata data node, carries the "real data" that we want to document/provide. The syntax and semantics of content-data is defined by the content-schema.

Two formats are specified based on the XML and JSON YANG encodings. Later as other YANG encodings (e.g. CBOR) are defined further instance data formats may be specified.

The content-data part SHALL follow the encoding rules defined in [RFC7950] for XML and [RFC7951] for JSON and MUST use UTF-8 character encoding. Content-data MAY include:

metadata as defined by [RFC7952].

a default attribute as defined in [RFC6243] section 6. and in [RFC8040] section 4.8.9.

origin metadata as specified in [RFC8526] and [RFC8527]

implementation specific metadata. Unknown metadata MUST be ignored by users of YANG instance data, allowing it to be used later for other purposes.

in the XML format implementation specific XML attributes. Unknown attributes MUST be ignored by users of YANG instance data, allowing them to be used later for other purposes.

The content-data part will be very similar to the result returned for a NETCONF <get-data> or for a RESTCONF get operation.

The content-data part MUST conform to the content-schema. An instance data set MAY contain data for any number of YANG modules; if needed it MAY carry the complete configuration and state data set for a server. Default values SHOULD NOT be included.

Config=true and config=false data MAY be mixed in the instance data file.

Instance data files MAY contain partial data sets. This means mandatory, min-elements, require-instance=true, must and when constrains MAY be violated.

The name of the instance data file SHOULD take one of the following two forms:

If revision information inside the data set is present

* instance-data-set-name ['@' revision-date] '.filetype'

* E.g. acme-router-modules@2018-01-25.xml

If the leaf name is present in the instance data header this MUST be used. Revision-date MUST be set to the latest revision date inside the instance data set.

If timestamp information inside the data set is present

* instance-data-set-name ['@' timestamp] '.filetype'

* E.g. acme-router-modules@2018-01-25T15_06_34_3+01_00.json

If the leaf name is present in the instance data header this MUST be used. If the leaf timestamp is present in the instance data header this MUST be used; the semicolons and the decimal point if present shall be replaced by underscores.

The revision date or timestamp is optional. ".filetype" SHALL be ".json" or ".xml" according to the format used.

Metadata, information about the data set itself SHOULD be included in the instance data set. Some metadata items are defined in the YANG module ietf-yang-instance-data, but other items MAY also be used. Metadata SHOULD include:

- Name of the data set
- Content schema specification
- Description of the instance data set. The description SHOULD contain information whether and how the data can change during the lifetime of the server.

3.1. Specifying the Content Schema

To properly understand and use an instance data set the user needs to know the content-schema. One of the following methods SHOULD be used:

Inline method: Include the needed information as part of the instance data set.

Simplified-Inline method: Include the needed information as part of the instance data set; short specification.

URI method: Include a URI that references another YANG instance data file. This instance data file will use the same content-schema as the referenced YANG instance data file. (if you don't want to repeat the info again and again)

EXTERNAL Method: Do not include the content-schema as it is already known, or the information is available through external documents.

Additional methods e.g. a YANG-package based solution may be added later.

Note, the specified content-schema only indicates the set of modules that were used to define this YANG instance data set. Sometimes instance data may be used for a server supporting a different YANG module set. (e.g. for "UC2 Preloading Data" the instance data set may not be updated every time the YANG modules on the server are updated) Whether the instance data set is usable for a possibly different real-life YANG module set depends on many factors including the compatibility between the specified and the real-life YANG module set (considering modules, revisions, features, deviations), the scope of the instance data, etc.

3.1.1. Inline Method

One or more inline-target-spec elements define YANG module(s) used to specify the content defining YANG modules.

E.g. ietf-yang-library@2016-06-21.yang

The anydata inline-content-schema carries instance data (conforming to the inline-target-spec modules) that actually specifies the content defining YANG modules including revision, supported features, deviations and any relevant additional data (e.g. version labels)

3.1.2. Simplified-Inline Method

The instance data set contains a list of content defining YANG modules including the revision date for each. Usage of this method implies that the modules are used without any deviations and with all features supported.

3.1.3. URI Method

A schema-uri leaf SHALL contain a URI that references another YANG instance data file. The current instance data file will use the same content schema as the referenced file.

The referenced instance data file MAY have no content-data if it is used solely for specifying the content-schema. The referenced YANG instance data file might use the INLINE method or might use the URI method to reference further instance data file(s). However at the

end of this reference chain there MUST be an instance data file using the INLINE method.

If a referenced instance data file is not available the revision data, supported features and deviations for the target YANG modules are unknown.

The URI method is advantageous when the user wants to avoid the overhead of specifying the content-schema in each instance data file: E.g. In Use Case 6, when the system creates a diagnostic file every minute to document the state of the server.

3.2. Examples

The following example is based on "UC1, Documenting Server Capabilities". It provides (a shortened) list of supported YANG modules and Netconf capabilities for a server. It uses the inline method to specify the content-schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-router-modules</name>
  <inline-spec>
    ietf-yang-library@2016-06-21.yang
  </inline-spec>
  <inline-content-schema>
    <module-state xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
      <module>
        <name>ietf-yang-library</name>
        <revision>2016-06-21</revision>
      </module>
      <module>
        <name>ietf-netconf-monitoring</name>
        <revision>2010-10-04</revision>
      </module>
    </module-state>
  </inline-content-schema>
  <revision>
    <date>1956-10-23</date>
    <description>Initial version</description>
  </revision>
  <description>Defines the minimal set of modules that any acme-router
      will contain.</description>
  <contact>info@acme.com</contact>
  <content-data>
    <!-- The example lists only 4 modules, but it could list the
      full set of supported modules for a server, potentially many
```

```
      dozens of modules -->
    <module-state xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
      <module>
        <name>ietf-yang-library</name>
        <revision>2016-06-21</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-yang-library
        </namespace>
        <conformance-type>implement</conformance-type>
      </module>
      <module>
        <name>ietf-system</name>
        <revision>2014-08-06</revision>
        <namespace>urn:ietf:params:xml:ns:yang:ietf-system</namespace>
        <feature>sys:authentication</feature>
        <feature>sys:local-users</feature>
        <deviation>
          <name>acme-system-ext</name>
          <revision>2018-08-06</revision>
        </deviation>
        <conformance-type>implement</conformance-type>
      </module>
      <module>
        <name>ietf-yang-types</name>
        <revision>2013-07-15</revision>
        <namespace>urn:ietf:params:xml:ns:yang:ietf-yang-types
          </namespace>
        <conformance-type>import</conformance-type>
      </module>
      <module>
        <name>acme-system-ext</name>
        <revision>2018-08-06</revision>
        <namespace>urn:rdns:acme.com:oammodel:acme-system-ext
          </namespace>
        <conformance-type>implement</conformance-type>
      </module>
    </module-state>
    <netconf-state>
      <capabilities>
        <capability>
          urn:ietf:params:netconf:capability:validate:1.1
        </capability>
      </capabilities>
    </netconf-state>
  </content-data>
</instance-data-set>
```

Figure 1: XML Instance Data Set - Use case 1, Documenting server capabilities

The following example is based on "UC2, Preloading Default Configuration". It provides a (shortened) default rule set for a read-only operator role. It uses the inline method for specifying the content-schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>read-only-acm-rules</name>
  <inline-spec>ietf-yang-library@2019-01-04.yang</inline-spec>
  <inline-content-schema>
    <yang-library xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
      <module-set>
        <name>all</name>
        <module>
          <name>ietf-netconf-acm</name>
          <revision>2012-02-22</revision>
        </module>
      </module-set>
    </yang-library>
  </inline-content-schema>
  <revision>
    <date>1776-07-04</date>
    <description>Initial version</description>
  </revision>
  <description>Access control rules for a read-only role.</description>
  <content-data>
    <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
      <enable-nacm>true</enable-nacm>
      <read-default>deny</read-default>
      <exec-default>deny</exec-default>
      <rule-list>
        <name>read-only-role</name>
        <group>read-only-group</group>
        <rule>
          <name>read-all</name>
          <module-name>*</module-name>
          <access-operation>read</access-operation>
          <action>permit</action>
        </rule>
      </rule-list>
    </nacm>
  </content-data>
</instance-data-set>
```

Figure 2: XML Instance Data Set - Use case 2, Preloading access control data

The following example is based on UC6 Storing diagnostics data. An instance data set is produced by the server every 15 minutes that contains statistics about NETCONF. As a new set is produced periodically many times a day a revision-date would be useless; instead a timestamp is included.

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "acme-router-netconf-diagnostics",
    "schema-uri": "file:///acme-netconf-diagnostics-yanglib.json",
    "timestamp": "2018-01-25T17:00:38Z",
    "description":
      "Netconf statistics",
    "content-data": {
      "ietf-netconf-monitoring:netconf-state": {
        "statistics": {
          "netconf-start-time ": "2018-12-05T17:45:00Z",
          "in-bad-hellos ": "32",
          "in-sessions ": "397",
          "dropped-sessions ": "87",
          "in-rpcs ": "8711",
          "in-bad-rpcs ": "408",
          "out-rpc-errors ": "408",
          "out-notifications": "39007"
        }
      }
    }
  }
}
```

Figure 3: JSON Instance Data File example - UC6 Storing diagnostics data

4. Data Life cycle

In UC2 "Preloading default configuration data" the loaded data may be changed later e.g. by management operations. In UC6 "Storing Diagnostics data" the diagnostics values may change on device every second.

YANG instance data is a snap-shot of information at a specific point of time. If the data changes afterwards this is not represented in the instance data set anymore. The valid values can be retrieved in run-time via NETCONF/RESTCONF or received e.g. in Yang-Push notifications.

Whether the instance data changes and if so, when and how, SHOULD be described either in the instance data set's description statement or in some other implementation specific manner.

5. Delivery of Instance Data

Instance data sets that are produced as a result of some sort of specification or design effort SHOULD be available without the need for a live server e.g. via download from the vendor's website, or in any other way product documentation is distributed.

Other instance data sets may be read from or produced by the YANG server itself e.g. UC6 documenting diagnostic data.

6. Backwards Compatibility

The concept of backwards compatibility and what changes are backwards compatible are not defined for instance data sets as it is highly dependent on the specific use case and the content-schema.

For instance data that is the result of a design or specification activity some changes that may be good to avoid are listed. YANG uses the concept of managed entities identified by key values; if the connection between the represented entity and the key value is not preserved during an update this may lead to problems.

- If the key value of a list entry that represents the same managed entity as before is changed, the user may mistakenly identify the list entry as new.

- If the meaning of a list entry is changed, but the key values are not (e.g. redefining an alarm-type but not changing its alarm-type-id) the change may not be noticed.

- If the key value of a previously removed list entry is reused for a different entity, the change may be mis-interpreted as reintroducing the previous entity.

7. Yang Instance Data Model

7.1. Tree Diagram

The following tree diagram [RFC8340] provides an overview of the data model.

```
   module: ietf-yang-instance-data
     structure instance-data-set:
        +--rw name?                          string
        +--rw (content-schema-spec)?
        |  +--:(simplified-inline)
        |     +--rw module*                  string
        |  +--:(inline)
        |  |  +--rw inline-spec*             string
        |  |  +--rw inline-content-schema    <anydata>
        |  +--:(uri)
        |     +--rw schema-uri?           inet:uri
        +--rw description?          string
        +--rw contact?              string
        +--rw organization?         string
        +--rw datastore?            ds:datastore-ref
        +--rw revision* [date]
        |  +--rw date           string
        |  +--rw description?   string
        +--rw timestamp?            yang:date-and-time
        +--rw content-data?         <anydata>
```

7.2. YANG Model

```
  <CODE BEGINS> file "ietf-yang-instance-data@2019-07-04.yang"
  module ietf-yang-instance-data {
    yang-version 1.1;
    namespace
      "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data";
    prefix yid ;

    import ietf-yang-structure-ext { prefix sx; }
    import ietf-datastores { prefix ds; }
    import ietf-inet-types { prefix inet; }
    import ietf-yang-types { prefix yang; }
    import ietf-yang-metadata { prefix "md"; }

    organization  "IETF NETMOD Working Group";
    contact
     "WG Web:   <https://datatracker.ietf.org/wg/netmodf/>
      WG List:  <mailto:netmod@ietf.org>

      Author:  Balazs Lengyel
         <mailto:balazs.lengyel@ericsson.com>";

    description "The module defines the structure and content of YANG
       instance data sets.

       The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
```

```
       'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
       'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
       are to be interpreted as described in BCP 14 (RFC 2119)
       (RFC 8174) when, and only when, they appear in all
       capitals, as shown here.

       Copyright (c) 2019 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; see
       the RFC itself for full legal notices.";

    revision 2019-07-04 {
      description  "Initial revision.";
      reference "RFC XXXX: YANG Instance Data Format";
    }

    sx:structure instance-data-set {
      description "A data structure to define a format for a
        YANG instance data set.Consists of meta-data about
        the instance data set and the real content-data.";

      leaf name {
        type string;
        description "Name of the YANG instance data set.";
      }

      choice content-schema-spec {
        description "Specification of the content-schema";

        case simplified-inline {
          leaf-list module  {
            type string {
              pattern '.+@\d{4}-\d{2}-\d{2}\.yang';
             }
             description "The list of content defining YANG
               modules including the revision date for each.
               Usage of this leaf-list implies the modules are
               used without any deviations and with all features
               supported.";
          }
```

```
        }
        case inline {
          leaf-list inline-spec {
            type string {
              pattern '.+@\d{4}-\d{2}-\d{2}\.yang';
            }
            min-elements 1;
            ordered-by user;
            description
              "Indicates that content defining Yang modules
              are specified inline.
              Each value MUST be a YANG Module name including the
              revision-date as defined for YANG file names in RFC7950.

              E.g. ietf-yang-library@2016-06-21.yang

              The first item is either ietf-yang-library or some other
              YANG module that contains a list of YANG modules with
              their name, revision-date, supported-features and
              deviations.
              As some versions of ietf-yang-library MAY contain
              different module-sets for different datastores, if
              multiple module-sets are included, the instance data set's
              meta-data MUST contain the datastore information and
              instance data for the ietf-yang-library MUST also contain
              information specifying the module-set for the relevant
              datastore.

              Subsequent items MAY specify YANG modules augmenting the
              first module with useful data (e.g. a version label).";
          }
          anydata inline-content-schema {
            mandatory true;
            description "Instance data corresponding to the YANG modules
              specified in the inline-spec nodes defining the set
              of content defining Yang YANG modules for this
              instance-data-set.";
          }
        }

        case uri {
          leaf schema-uri {
            type inet:uri;
            description
              "A reference to another YANG instance data file.
              This instance data file will use the same set of target
              YANG modules, revisions, supported features and deviations
              as the referenced YANG instance data file.";
```

```
        }
      }
    }

    leaf-list description {
      type string;
      description "Description of the instance data set.";
    }

    leaf contact {
      type string;
      description "Contact information for the person or
        organization to whom queries concerning this
        instance data set should be sent.";
    }

    leaf organization {
      type string;
      description "Organization responsible for the instance
        data set.";
    }

    leaf datastore {
      type ds:datastore-ref;
      description  "The identity of the datastore with which the
        instance data set is associated e.g. the datastore from
        where the data was read or the datastore where the data
        could be loaded or the datastore which is being documented.
        If a single specific datastore can not be specified, the
        leaf MUST be absent.

        If this leaf is absent, then the datastore to which the
        instance data belongs is undefined.";
    }

    list revision {
      key date;
      description "Instance data sets that are produced as
        a result of some sort of specification or design effort
        SHOULD have at least one revision entry.  For every
        published editorial change, a new one SHOULD be added
        in front of the revisions sequence so that all
        revisions are in reverse chronological order.

        For instance data sets that are read from
        or produced by a server or otherwise
        subject to frequent updates or changes, revision
        SHOULD NOT be present";
```

```
        leaf date {
          type string {
            pattern '\d{4}-\d{2}-\d{2}';
          }
          description "Specifies the date the instance data set
            was last modified. Formatted as YYYY-MM-DD";
        }

        leaf description {
          type string;
          description
            "Description of this revision of the instance data set.";
        }
      }

      leaf timestamp {
        type yang:date-and-time;
        description "The date and time when the instance data set
          was last modified.

          For instance data sets that are read from or produced
          by a server or otherwise subject to frequent
          updates or changes, timestamp SHOULD be present";
      }

      anydata content-data {
        description "Contains the real instance data.
          The data MUST conform to the relevant YANG Modules specified
          either in the content-schema-spec or in some other
          implementation specific manner.";
      }
    }
  }
  <CODE ENDS>
```

8. Security Considerations

Depending on the nature of the instance data, instance data files MAY need to be handled in a secure way. The same type of handling should be applied, that would be needed for the result of a <get> operation returning the same data.

9. IANA Considerations

This document registers one URI and one YANG module.

9.1. URI Registration

This document registers one URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-yang-instance-data

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

9.2. YANG Module Name Registration

This document registers one YANG module in the YANG Module Names registry [RFC6020].

name: ietf-yang-instance-data
namespace: urn:ietf:params:xml:ns:yang:ietf-yang-instance-data
prefix: yid
reference: RFC XXXX

10. Acknowledgments

For their valuable comments, discussions, and feedback, we wish to acknowledge Andy Bierman, Juergen Schoenwaelder, Rob Wilton, Joe Clarke, Kent Watsen Martin Bjorklund, Ladislav Lhotka, Qin Wu and other members of the Netmod WG.

11. References

11.1. Normative References

[I-D.ietf-netmod-yang-data-ext]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Data Structure Extensions", draft-ietf-netmod-yang-data-ext-04 (work in progress), July 2019.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6243] Bierman, A. and B. Lengyel, "With-defaults Capability for NETCONF", RFC 6243, DOI 10.17487/RFC6243, June 2011, <https://www.rfc-editor.org/info/rfc6243>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <https://www.rfc-editor.org/info/rfc7952>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8526] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "NETCONF Extensions to Support the Network Management Datastore Architecture", RFC 8526, DOI 10.17487/RFC8526, March 2019, <https://www.rfc-editor.org/info/rfc8526>.

[RFC8527] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "RESTCONF Extensions to Support the Network Management Datastore Architecture", RFC 8527, DOI 10.17487/RFC8527, March 2019, <https://www.rfc-editor.org/info/rfc8527>.

11.2. Informative References

[I-D.ietf-ccamp-alarm-module] Vallin, S. and M. Bjorklund, "YANG Alarm Module", draft-ietf-ccamp-alarm-module-09 (work in progress), April 2019.

[I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-07 (work in progress), October 2018.

[I-D.ietf-netconf-yang-push]
Clemm, A. and E. Voit, "Subscription to YANG Datastores", draft-ietf-netconf-yang-push-25 (work in progress), May 2019.

[I-D.wu-netconf-restconf-factory-restore]
Wu, Q., Lengyel, B., and Y. Niu, "Factory default Setting", draft-wu-netconf-restconf-factory-restore-03 (work in progress), October 2018.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

Appendix A. Open Issues

- -

Appendix B. Changes between revisions

v03 - v04

- removed entity-tag and last-modified timestamp
- Added simplified-inline method of content-schema specification

v02 - v03

- target renamed to "content-schema" and "content defining Yang module(s)"
- Made name of instance data set optional
- Updated according to draft-ietf-netmod-yang-data-ext-03
- Clarified that entity-tag and last-modified timestamp are encoded as metadata. While they contain useful data, the HTTP-header based encoding from Restconf is not suitable.

v01 - v02

- Removed design time from terminology
- Defined the format of the content-data part by referencing various RFCs and drafts instead of the result of the get-data and get operations.
- Changed target-ptr to a choice
- Inline target-ptr may include augmenting modules and alternatives to ietf-yang-library
- Moved list of target modules into a separate <target-modules> element.
- Added backwards compatibility considerations

v00 - v01

- Added the target-ptr metadata with 3 methods
- Added timestamp metadata
- Removed usage of dedicated .yid file extension
- Added list of use cases
- Added list of principles
- Updated examples
- Moved detailed use case descriptions to appendix

Appendix C. Detailed Use Cases - Non-Normative

C.1. Use Cases

We present a number of use cases were YANG instance data is needed.

C.1.1. Use Case 1: Early Documentation of Server Capabilities

A server has a number of server-capabilities that are defined in YANG modules and can be retrieved from the server using protocols like NETCONF or RESTCONF. server capabilities include

- data defined in ietf-yang-library: YANG modules, submodules, features, deviations, schema-mounts, datastores supported ([I-D.ietf-netconf-rfc7895bis])

- alarms supported ([I-D.ietf-ccamp-alarm-module])

- data nodes, subtrees that support or do not support on-change notifications ([I-D.ietf-netconf-yang-push])

- netconf-capabilities in ietf-netconf-monitoring

While it is good practice to allow a client to query these capabilities from the live server, that is often not possible.

Often when a network node is released an associated NMS (network management system) is also released with it. The NMS depends on the capabilities of the server. During NMS implementation information about server capabilities is needed. If the information is not available early in some off-line document, but only as instance data from the live network node, the NMS implementation will be delayed, because it has to wait for the network node to be ready. Also assuming that all NMS implementors will have a correctly configured network node available to retrieve data from, is a very expensive proposition. (An NMS may handle dozens of node types.)

Network operators often build their own home-grown NMS systems that needs to be integrated with a vendor's network node. The operator needs to know the network node's server capabilities in order to do this. Moreover the network operator's decision to buy a vendor's product may even be influenced by the network node's OAM feature set documented as the Server's capabilities.

Beside NMS implementors, system integrators and many others also need the same information early. Examples could be model driven testing, generating documentation, etc.

Most server-capabilities are relatively stable and change only during upgrade or due to licensing or addition or removal of HW. They are usually defined by a vendor at design time, before the product is released. It feasible and advantageous to define/document them early e.g. in a YANG instance data File.

It is anticipated that a separate IETF document will define in detail how and which set of server capabilities should be documented.

C.1.2. Use Case 2: Preloading Data

There are parts of the configuration that must be fully configurable by the operator, however for which often a simple default configuration will be sufficient.

One example is access control groups/roles and related rules. While a sophisticated operator may define dozens of different groups often a basic (read-only operator, read-write system administrator, security-administrator) triplet will be enough. Vendors will often provide such default configuration data to make device configuration easier for an operator.

Defining Access control data is a complex task. To help the device vendor pre-defines a set of default groups (/nacm:nacm/groups) and rules for these groups to access specific parts of common models (/nacm:nacm/rule-list/rule).

YANG instance data files are used to document and/or preload the default configuration.

C.1.3. Use Case 3: Documenting Factory Default Settings

Nearly every server has a factory default configuration. If the system is really badly misconfigured or if the current configuration is to be abandoned the system can be reset to this default.

In Netconf the <delete-config> operation can already be used to reset the startup datastore. There are ongoing efforts to introduce a new, more generic reset-datastore operation for the same purpose [I-D.wu-netconf-restconf-factory-restore]

The operator currently has no way to know what the default configuration actually contains. YANG instance data can be used to document the factory default configuration.

Authors' Addresses

Balazs Lengyel
Ericsson
Magyar Tudosok korutja 11
1117 Budapest
Hungary

Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Phone: +32 2 704 5622
Email: bclaise@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 4, 2020

J. Clarke, Ed.
Cisco Systems, Inc.

July 3, 2019

YANG Module Versioning Requirements
draft-ietf-netmod-yang-versioning-reqs-01

Abstract

This document describes the problems that can arise because of the YANG language module update rules, that require all updates to YANG module preserve strict backwards compatibility. It also defines the requirements on any solution designed to solve the stated problems. This document does not consider possible solutions, nor endorse any particular solution.

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 https://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, 2020.

Copyright Notice

Copyright (c) 2019 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 (https://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 2
2.1. Striving for model perfection 3
2.2. Some YANG Modules Are Not Backwards-Compatible 3
2.3. Non-Backwards-Compatible Errors 4
2.4. No way to easily decide whether a change is Backwards-Compatible 4
2.5. No good way to specify which module revision to import . 5
2.6. Early Warning about Removal 6
2.7. Clear Indication of Node Support 6
3. Terminology and Conventions 7
4. The Problem Statement 7
5. Requirements of a YANG Versioning Solution 9
6. Contributors 11
7. Acknowledgments 11
8. Security Considerations 11
9. IANA Considerations 12
10. References 12
10.1. Normative References 12
10.2. Informative References 12
Author's Address 12

1. Introduction

This requirements document initially considers some of the existing YANG module update rules, then describes the problems that arise due to those rules embracing strict backwards compatibility, and finally defines requirements on any solution that may be designed to solve these problems by providing an alternative YANG versioning strategy.

2. Background

The YANG data modeling language [RFC7950] specifies strict rules for updating YANG modules (see section 11 "Updating a Module"). Citing a few of the relevant rules:

1. "As experience is gained with a module, it may be desirable to revise that module. However, changes to published modules are not allowed if they have any potential to cause interoperability problems between a client using an original specification and a server using an updated specification."

2. "Note that definitions contained in a module are available to be imported by any other module and are referenced in "import" statements via the module name. Thus, a module name MUST NOT be changed. Furthermore, the "namespace" statement MUST NOT be changed, since all XML elements are qualified by the namespace."

3. "Otherwise, if the semantics of any previous definition are changed (i.e., if a non-editorial change is made to any definition other than those specifically allowed above), then this MUST be achieved by a new definition with a new identifier."

4. "deprecated indicates an obsolete definition, but it permits new/ continued implementation in order to foster interoperability with older/existing implementations."

The rules described above, along with other similar rules, causes various problems, as described in the following sections:

2.1. Striving for model perfection

The points made above lead to the logical conclusion that the standardized YANG modules have to be perfect on day one (at least the structure and meaning), which in turn might explain why IETF YANG modules take so long to standardize. Shooting for perfection is obviously a noble goal, but if the perfect standard comes too late, it doesn't help the industry.

2.2. Some YANG Modules Are Not Backwards-Compatible

As we learn from our mistakes, we're going to face more and more non-backwards-compatible YANG modules. An example is the YANG data model for L3VPN service delivery [RFC8049], which, based on implementation experience, has been updated in a non-backwards-compatible way by [RFC8299].

While Standards Development Organization (SDO) YANG modules are obviously better for the industry, we must recognize that many YANG modules are actually generated YANG modules (for example, from internal databases), which is sometimes the case for vendor modules [RFC8199]. From time to time, the new YANG modules are not backwards-compatible.

Old module parts that are no longer needed, no longer supported, or are not used by consumers need to be removed from modules. It is often hard to decide which parts are no longer needed/used; still the need and practice of removing old parts exist. While it is rare in standard modules it is more common in vendor YANG modules where the usage of modules is more controlled.

The problems described in Section 2.7 may also result in incompatible changes.

In such cases, it would be better to indicate how backwards-compatible a given YANG module actually is.

As modules are sometimes updated in an incompatible way the current assumption that once a YANG module is defined all further revisions can be freely used as they are compatible is not valid.

2.3. Non-Backwards-Compatible Errors

Sometimes small errors force us to make non-backwards-compatible updates. As an example imagine that we have a string with a complex pattern (e.g., an IP address). Let's assume the initial pattern incorrectly allows IP addresses to start with 355. In the next version this is corrected to disallow addresses starting with 355. Formally this is a non-backwards-compatible change as the value space of the string is decreased. In reality an IP address and the implementation behind it was never capable of handling an address starting with 355. So practically this is a backwards-compatible change, just like a correction of the description statement. Current YANG rules are ambiguous as to whether non-backwards-compatible bug fixes are allowed without also requiring a module name change.

2.4. No way to easily decide whether a change is Backwards-Compatible

A management system, SDN controller, or any other user of a module should be capable of easily determining the compatibility between two module versions. Higher level logic for a network function, something that cannot be implemented in a purely model driven way, is always dependent on a specific version of the module. If the client finds that the module has been updated on the network node, it has to decide if it tries to handle it as it handled the previous version of the model or if it just stops, to avoid problems. To make this decision the client needs to know if the module was updated in a backwards-compatible way or not.

This is not possible to decide today because of the following:

- It is sometimes necessary to change the semantic behavior of a data node, action or rpc while the YANG definition does not change (with the possible exception of the description statement). In such a case it is impossible to determine whether the change is backwards-compatible just by looking at the YANG statements. It's only the human model designer who can decide.

- Problems with the deprecated and obsolete status statement, Section 2.7

- YANG module authors might decide to violate YANG 1.1 update rules for some of the reasons above.

Finding status changes or violations of update rules need a line-by-line comparison of the old and new modules is a tedious task.

2.5. No good way to specify which module revision to import

If a module (MOD-A) is imported by another one (MOD-B) the importer may specify which revision must be imported. Even if MOD-A is updated in a backwards-compatible way not all revisions will be suitable, e.g., a new MOD-B might need the newest MOD-A. However, both specifying or omitting the revision date for import leads to problems.

If the import by revision-date is specified

- If corrections are made to MOD-A these would not have any effect as the import's revision date would still point to the un-corrected earlier YANG module revision.

- If MOD-A is updated in a backwards-compatible way because another importer (MOD-C) needs some functionality, the new MOD-A could be used by MOD-B, but specifying the exact import revision-date prevents this. This will force the implementers to import two different revisions of MOD-A, forcing them to maintain old MOD-A revisions unnecessarily.

- If multiple modules import different revisions of MOD-A the human user will need to understand the subtle differences between the different revisions. Small differences would easily lead to operator mistakes as the operator will rarely check the documentation.

- Tooling/SW is often not prepared to handle multiple revisions of the same YANG module.

If the import revision-date is not specified

- any revision of MOD-A may be used including unsuitable ones. Older revisions may be lacking functionality MOD-B needs. Newer MOD-A revisions may obsolete definitions used by MOD-B in which case these must not be used by MOD-B anymore.

- As it is not specified which revisions of MOD-A are suitable for MOD-B. The problem has to be solved on a case by case basis studying all the details of MOD-A and MOD-B which is considerable work.

2.6. Early Warning about Removal

If a schema part is considered old/bad we need to be able to give advance warning that it will be removed. As this is an advance warning the part must still be present and usable in the current revision; however, it will be removed in one of the next revisions. The deprecated statement cannot be reliably used for this purpose both because deprecated nodes may not be implemented and also there is no mandate that text be provided explaining the deprecation.

We need the advance warning to allow users of the module time to plan/execute migration away from the deprecated functionality. Deprecation should be accompanied by information whether the functionality will just disappear or that there is an alternative, possibly more advanced solution that should be used.

Vendors use such warnings often, but the NMDA related redesign of IETF modules is also an example where it would be useful for IETF. As another example, see the usage of deprecated in the Java programming language.

2.7. Clear Indication of Node Support

The current definition of deprecated and obsolete in [RFC7950] (as quoted below) is problematic and should be corrected.

- "deprecated" indicates an obsolete definition, but it permits new/ continued implementation in order to foster interoperability with older/existing implementations.

- "obsolete" means that the definition is obsolete and SHOULD NOT be implemented and/or can be removed from implementations.

YANG is considered an interface contract between the server and the client. The current definitions of deprecated and obsolete mean that a schema node that is either deprecated or obsolete may or may not be implemented. The client has no way to find out which is the case except for by trying to write or read data at the leaf in question. This probing would need to be done for each separate data-node, which is not a trivial thing to do. This "may or may not" is unacceptable in a contract. In effect, this works as if there would be an if-feature statement on each deprecated schema node where the server

does not advertise whether the feature is supported or not. Why is it not advertised?

3. Terminology and Conventions

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

In addition, this document uses the following terminology:

- YANG module revision: An instance of a YANG module, with no implied ordering or backwards compatibility between different revisions of the same module."

- YANG module version: A YANG module revision, but also with an implied partial ordering relationship between other versions of the same module. Each module version must be uniquely identifiable.

- Non-backwards-compatible (NBC): In the context of this document, the term 'non-backwards-compatible' refers to a change or set of changes between two YANG module revisions that do not adhere to the list of allowable changes specified in Section 11 "Updating a Module" of [RFC7950], with the following additional clarification:

  * Any addition of, or change to, a "status" statement that allows a server to remove support for a schema node is considered a non-backwards-compatible change

4. The Problem Statement

Considering the issues described in the background, the problem definition can be summarized as follows.

Development of data models for a large collection of communication protocols and system components is difficult and typically only manageable with an iterative development process. Agile development approaches advocate evolutionary development, early delivery, and continual improvement. They are designed to support rapid and flexible response to change. Agile development has been found to be very successful in a world where the objects being modeled undergo constant changes.

The current module versioning scheme relies on the fundamental idea that a definition, once published, never changes its semantics. As a consequence, if a new definition is needed with different non-backwards-compatible semantics, then a new definition must be created

to replace the old definition. The advantage of this versioning scheme is that a definition identified by a module name and a path has fixed semantics that never change. (The details are a bit more nuanced but we simplify things here a bit in order to get the problems worked out clearly.)

There are two main disadvantages of the current YANG versioning scheme:

- Any non-backwards-compatible change of a definition requires either a new module name or a new path. This has been found costly to support in implementations, in particular on the client side.

- Since non-backwards-compatible changes require either a new module name or a new path, such changes will impact other modules that import definitions. In fact, with the current module versioning scheme other modules have to opt-in in order to use the new version. This essentially leads to a ripple effect where a non-backwards-compatible change of a core module causes updates on a potentially large number of dependent modules.

Other problems experienced with the current YANG versioning scheme are the following:

- YANG has a mechanism to mark definitions deprecated but it leaves it open whether implementations are expected to implement deprecated definitions and there is no way (other than trial and error) for a client to find out whether deprecated definitions are supported by a given implementation.

- YANG does not have a robust mechanism to document which data definitions have changed and to provide guidance how implementations should deal with the change. While it is possible to have this described in general description statements, having these details embedded in general description statements does not make this information accessible to tools.

- YANG data models often do not exist in isolation and they interact with other software systems or data models that often do allow (controlled) non-backwards-compatible changes. In some cases, YANG models are mechanically derived from other data models that do allow (controlled) non-backwards-compatible changes. In such situations, a robust mapping to YANG requires to have version numbers exposed as part of the module name or a path definition, which has been found to be expensive on the client side (see above).

Given the need to support agile development processes and the disadvantages and problems of the current YANG versioning scheme described above, it is necessary to develop requirements and solutions for a future YANG versioning scheme that better supports agile development processes, whilst retaining the ability for servers to handle clients using older versions of YANG modules.

5. Requirements of a YANG Versioning Solution

The following is a list of requirements that a solution to the problems mentioned above MUST or SHOULD have. The list is grouped by similar requirements but is not presented in a set priority order.

1. Requirements related to making non-backwards-compatible updates to modules:

    1.1 A mechanism is REQUIRED to update a module in a non-backwards-compatible way without forcing all modules with import dependencies on the updated module from being updated at the same time (e.g. to change its import to use a new module name).

    1.2 Non-backwards-compatible updates of a module MUST not impact clients that only access data nodes of the module that have either not been updated or have been updated in backwards-compatible ways.

    1.3 A refined form of YANG's 'import' statement MUST be provided that is more restrictive than "import any revision" and less restrictive than "import a specific revision". Once non-backwards-compatible changes to modules are allowed, the refined import statement is used to express the correct dependency between modules.

    1.4 The solution MUST be able to express when non-backwards-compatible changes have occurred between two revisions of a given YANG module.

2. Requirements related to identifying changes between different module revisions:

    2.1 Readers of modules, and tools that use modules, MUST be able to determine whether changes between two revisions of a module constitute a backwards-compatible or non-backwards-compatible version change. In addition, it MAY be helpful to identify whether changes represent bug fixes, new functionality, or both.

2.2 A mechanism SHOULD be defined to determine whether data nodes between two arbitrary YANG module revisions have (i) not changed, (ii) changed in a backwards-compatible way, (iii) changed in a non-backwards-compatible way.

3. Requirements related to supporting existing clients in a backwards-compatible way:

3.1 The solution MUST provide a mechanism to allow servers to support existing clients in a backwards-compatible way.

3.2 The solution MUST provide a mechanism to support clients that expect an older version of a given module when the current version has had non-backwards-compatible changes.

3.3 Clients are expected to be able to handle unexpected instance data resulting from backwards-compatible changes.

4. Requirements related to managing and documenting the life cycle of data nodes:

4.1 A mechanism is REQUIRED to allow a client to determine whether deprecated nodes are implemented by the server.

4.2 If a data node is deprecated or obsolete then it MUST be possible to document in the YANG module what alternatives exist, the reason for the status change, or any other status related information.

4.3 A mechanism is REQUIRED to indicate that certain definitions in a YANG module will become status obsolete in future revisions but definitions marked as such MUST still be implemented by compliant servers.

5. Requirements related to documentation and education:

5.1 The solution MUST provide guidance to model authors and clients on how to use the new YANG versioning scheme.

5.2 The solution is REQUIRED to describe how to transition from the existing YANG 1.0/1.1 versioning scheme to the new scheme.

5.3 The solution MUST describe how the versioning scheme affects the interpretation of instance data and references to instance data, for which the schema definition has been updated in a non-backwards-compatible way.

6. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The following people are members of that design team and have contributed to defining the problem and specifying the requirements:

- Balazs Lengyel
- Benoit Claise
- Ebben Aries
- Jason Sterne
- Joe Clarke
- Juergen Schoenwaelder
- Mahesh Jethanandani
- Michael (Wangzitao)
- Qin Wu
- Reshad Rahman
- Rob Wilton

7. Acknowledgments

The design team would like to thank Christian Hopps and Vladimir Vassilev for their feedback and perspectives in shaping and fine tuning the versioning requirements.

One of the inspirations for solving the YANG module versioning comes from OpenConfig. The authors would like to thank Anees Shaikh and Rob Shakir for their helpful input.

8. Security Considerations

The document does not define any new protocol or data model. There is no security impact.

9. IANA Considerations

None

10. References

10.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

10.2. Informative References

[RFC8049] Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8049, DOI 10.17487/RFC8049, February 2017, <https://www.rfc-editor.org/info/rfc8049>.

[RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <https://www.rfc-editor.org/info/rfc8199>.

[RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <https://www.rfc-editor.org/info/rfc8299>.

Author's Address

Joe Clarke (editor)
Cisco Systems, Inc.
7200-12 Kit Creek Rd
Research Triangle Park, North Carolina
United States of America

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2020

R. Wilton
Cisco Systems, Inc.

October 23, 2019

YANG Packages
draft-rwilton-netmod-yang-packages-02

Abstract

This document defines YANG packages, a versioned organizational structure holding a set of related YANG modules, that collectively define a YANG schema. It describes how YANG instance data documents are used to define YANG packages, and how the YANG library information published by a server can be augmented with packages related information.

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 https://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 25, 2020.

Copyright Notice

Copyright (c) 2019 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 (https://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. Terminology and Conventions 3
2. Introduction 4
3. Background on YANG packages 4
4. Objectives 5
5. YANG Package Definition 6
5.1. Package definition rules 7
5.2. Package versioning 7
5.2.1. Updating a package with a new version 8
5.2.1.1. Non-Backwards-compatible changes 8
5.2.1.2. Backwards-compatible changes 8
5.2.1.3. Editorial changes 9
5.2.2. YANG Semantic Versioning for packages 9
5.2.3. Revision history 10
5.3. Package conformance 10
5.3.1. Use of YANG semantic versioning 10
5.3.2. Package checksums 11
5.4. Schema referential completeness 11
5.5. Package name scoping and uniqueness 12
5.5.1. Globally scoped packages 12
5.5.2. Server scoped packages 12
5.6. Submodules packages considerations 13
5.7. Package tags 13
6. YANG Packages instance data 13
7. YANG Packages additions to YANG library 15
7.1. Package List 15
7.2. Binding from schema to package 15
7.3. Tree diagram 16
8. YANG Packages Groupings 18
9. YANG packages as schema for YANG instance data document . . . 18
10. YANG Modules 19
11. Security Considerations 40
12. IANA Considerations 41
13. Open Questions/Issues 42
14. Acknowledgements 42
15. References 42
15.1. Normative References 42
15.2. Informative References 44
Appendix A. Tree output for ietf-yang-library with package augmentations 45
Appendix B. Examples 47
B.1. Example IETF Network Device YANG package 47
B.2. Example IETF Basic Routing YANG package 50
B.3. Package import conflict resolution example 53

Appendix C. Possible alternative solutions 56
C.1. Using module tags 56
C.2. Using YANG library 56
Author's Address 57

1. Terminology and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses terminology introduced in the YANG versioning requirements draft [I-D.verdt-netmod-yang-versioning-reqs].

This document also makes of the following terminology introduced in the Network Management Datastore Architecture [RFC8342]:

- datastore schema

This document also makes of the following terminology introduced in the YANG 1.1 Data Modeling Language [RFC7950]:

- data node

In addition, this document defines the following terminology:

- YANG schema: A datastore schema, not bound to any particular datastore.

- YANG package: An organizational structure holding a collection of YANG modules related by some common purpose, normally defined in a YANG instance data file. A YANG package defines a YANG schema by specifying a set of YANG modules revisions, package versions, mandatory features, and deviations. YANG packages are defined in Section 5.

- backwards-compatible (BC) change: When used in the context of a YANG module, it follows the definition in Section 3.1.1 of [I-D.verdt-netmod-yang-module-versioning]. When used in the context of a YANG package, it follows the definition in Section 5.2.1.2.

- non-backwards-compatible (NBC) change: When used in the context of a YANG module, it follows the definition in Section 3.1.2 of [I-D.verdt-netmod-yang-module-versioning]. When used in the

context of a YANG package, it follows the definition in Section 5.2.1.2.

- editorial change: When used in the context of a YANG module, it follows the definition of an 'editorial change' in 3.2 of [I-D.verdt-netmod-yang-semver]. When used in the context of a YANG package, it follows the definition in Section 5.2.1.3.

2. Introduction

This document defines and describes the YANG [RFC7950] constructs that are used to define and use YANG packages.

A YANG package is an organizational structure that groups a set of YANG modules together into a consistent versioned definition to serve a common purpose. For example, a YANG package could define the set of YANG modules required to implement an L2VPN service on a network device. YANG packages can themselves refer to, and reuse, other package definitions.

Non-normative examples of YANG packages are provided in the appendices.

3. Background on YANG packages

It has long been acknowledged within the YANG community that network management using YANG requires a unit of organization and conformance that is broader in scope than individual YANG modules.

'The YANG Package Statement' [I-D.bierman-netmod-yang-package] proposed a YANG package mechanism based on new YANG language statements, where a YANG package is defined in a file similar to how YANG modules are defined, and would require enhancements to YANG compilers to understand the new statements used to define packages.

OpenConfig [openconfigsemver] describes an approach to versioning 'bundle releases' based on git tags. I.e. a set of modules, at particular versions, can be marked with the same release tag to indicate that they are known to interoperate together.

The NETMOD WG in general, and the YANG versioning design team in particular, are exploring solutions [I-D.verdt-netmod-yang-solutions] to the YANG versioning requirements, [I-D.verdt-netmod-yang-versioning-reqs]. Solutions to the versioning requirements can be split into several distinct areas. [I-D.verdt-netmod-yang-module-versioning] is focused on YANG versioning scoped to individual modules. The overall solution must also consider YANG versioning and conformance scoped to YANG schema.

YANG packages provide part of the solution for versioning YANG schema.

4. Objectives

The main goals of YANG package definitions include, but are not restricted to:

- To provide an alternative, simplified, YANG conformance mechanism. Rather than conformance being performed against a set of individual YANG module revisions, features, and deviations, conformance can be more simply stated in terms of YANG packages, with a set of modifications (e.g. additional modules, deviations, or features).

- To allow YANG schema to be specified in a concise way rather than having each server explicitly list all modules, revisions, and features. YANG package definitions can be defined in documents that are available offline, and accessible via a URL, rather than requiring explicit lists of modules to be shared between client and server. Hence, a YANG package must contain sufficient information to allow a client or server to precisely construct the schema associated with the package.

- To define a mainly linear versioned history of sets of modules versions that are known to work together. I.e. to help mitigate the problem where a client must manage devices from multiple vendors, and vendor A implements version 1.0.0 of module foo and version 2.0.0 of module bar, and vendor B implements version 2.0.0 of module foo and version 1.0.0 of module bar. For a client, trying to interoperate with multiple vendors, and many YANG modules, finding a consistent lowest common denominator set of YANG module versions may be difficult, if not impossible.

Protocol mechanisms of how clients can negotiate which packages or package versions are to be used for NETCONF/RESTCONF communications are outside the scope of this document, and are defined in [I-D.wilton-netmod-yang-ver-selection].

Finally, the package definitions proposed by this document are intended to be relatively basic in their definition and the functionality that they support. As industry gains experience using YANG packages, the standard YANG mechanisms of updating, or augmenting, YANG modules could also be used to extend the functionality supported by YANG packages, if required.

5. YANG Package Definition

This document specifies an approach to defining YANG packages that is different to either of the approaches described in the background.

A YANG package is a versioned organizational structure defining a set of related YANG modules, packages, features, and deviations. A YANG package collectively defines a YANG schema.

Each YANG package has a name that SHOULD end with the suffix "-pkg". Package names are normally expected to be globally unique, but in some cases the package name may be locally scoped to a server or device, as described in Section 5.5.

YANG packages are versioned using the same approaches described in [I-D.verdt-netmod-yang-module-versioning] and [I-D.verdt-netmod-yang-semver]. This is described in further detail in Section 5.2.

Each YANG package version, defines:

- some metadata about the package, e.g., description, tags, scoping, referential completeness, location information.
- a set of YANG modules, at particular revisions, that are implemented by servers that implement the package. The modules may contain deviations.
- a set of import-only YANG modules, at particular revisions, that are used 'import-only' by the servers that implement the package.
- a set of included YANG packages, at particular revisions, that are also implemented by servers that implement the package.
- a set of YANG module features that must be supported by servers that implement the package.

The structure for YANG package definitions uses existing YANG language statements, YANG Data Structure Extensions [I-D.ietf-netmod-yang-data-ext], and YANG Instance Data File Format [I-D.ietf-netmod-yang-instance-file-format].

YANG package definitions are available offline in YANG Instance Data Documents. Client applications can be designed to support particular package versions that they expect to interoperate with.

YANG package definitions are available from the server, via augmentations to YANG Library [RFC8525]. Rather than client

applications downloading the entire contents of YANG library to confirm that the server schema is compatible with the client, they can check, or download, a much shorter YANG package definition, and validate that it conforms to the expected schema.

YANG package definitions can also be used to define the schema associated with YANG instance data documents holding other, e.g., non packages related, instance data.

5.1. Package definition rules

The following rules define how packages are defined:

A YANG package MAY represent a complete YANG schema or only part of a YANG schema with some module import dependencies missing, as described in Section 5.4.

Packages definitions are hierarchical. A package can include other packages. Only a single version of a package can be included, and conflicting package includes (e.g. from descendant package includes) MUST be explicitly resolved by indicating which version takes precedence, and which versions are being replaced.

For each module implemented by a package, only a single revision of that module MUST be implemented. Multiple revisions of a module MAY be listed as import-only dependencies.

The revision of a module listed in the package 'module' list supersedes any 'implemented' revision of the module listed in an included package module list. The 'replaces-revision' leaf-list is used to indicate which 'implemented' or 'import-only' module revisions are replaces by this module revision. This allows a package to explicitly resolve conflicts between implemented module revisions in included packages.

The 'replaces-revision' leaf-list in the 'import-only-module' list can be used to exclude duplicate revisions of import-only modules from included packages. Otherwise, the import-only-modules for a package are the import-only-modules from all included packages combined with any modules listed in the packages import-only-module list.

5.2. Package versioning

Individual versions of a YANG package are versioned using the "revision-label" scheme defined in section 3.3 of [I-D.verdt-netmod-yang-module-versioning].

5.2.1. Updating a package with a new version

Package compatibility is fundamentally defined by how the YANG schema between two package versions has changed.

When a package definition is updated, the version associated with the package MUST be updated appropriately, taking into consideration the scope of the changes as defined by the rules below.

A package definition SHOULD define the previous version of the package in the 'previous-version' leaf unless it is the initial version of the package. If the 'previous-version' leaf is provided then the package definition MUST set the 'nbc-changes' leaf if the new version is non-backwards-compatible with respect to the package version defined in the 'previous-version' leaf.

5.2.1.1. Non-Backwards-compatible changes

The following changes classify as NBC changes to a package definition:

Changing an 'included-package' list entry to select a package version that is non-backwards-compatible to the prior package version, or removing a previously included package.

Changing a 'module' or 'import-only-module' list entry to select a module revision that is non-backwards-compatible to the prior module revision, or removing a previously implemented module.

Removing a feature from the 'mandatory-feature' leaf-list.

Adding, changing, or removing a deviation that is considered a non-backwards-compatible change to the affected data node in the schema associated with the prior package version.

5.2.1.2. Backwards-compatible changes

The following changes classify as BC changes to a package definition:

Changing an 'included-package' list entry to select a package version that is backwards-compatible to the prior package version, or including a new package that does not conflict with any existing included package or module.

Changing a 'module' or 'import-only-module' list entry to select a module revision that is backwards-compatible to the prior module revision, or including a new module to the package definition.

Adding a feature to the 'mandatory-feature' leaf-list.

Adding, changing, or removing a deviation that is considered a backwards-compatible change to the affected data node in the schema associated with the prior package version.

5.2.1.3. Editorial changes

The following changes classify as editorial changes to a package definition:

Changing a 'included-package' list entry to select a package version that is classified as an editorial change relative to the prior package version.

Changing a 'module' or 'import-only-module' list entry to select a module revision that is classified as an editorial change relative to the prior module revision.

Any change to any metadata associated with a package definition that causes it to have a different checksum value.

5.2.2. YANG Semantic Versioning for packages

YANG Semantic Versioning [I-D.verdt-netmod-yang-semver] MAY be used as an appropriate type of revision-label for the package version leaf.

If the format of the leaf matches the 'yangver:version' type specified in ietf-yang-semver.yang, then the package version leaf MUST be interpreted as a YANG semantic version number.

For YANG packages defined by the IETF, YANG semantic version numbers MUST be used as the version scheme for YANG packages.

The rules for incrementing the YANG package version number are equivalent to the semantic versioning rules used to version individual YANG modules, defined in section 3.2 of [I-D.verdt-netmod-yang-semver], but use the rules defined previously in Section 5.2.1 to determine whether a change is classified as non-backwards-compatible, backwards-compatible, or editorial. Where available, the semantic version number of the referenced elements in the package (included packages or modules) can be used to help determine the scope of changes being made.

5.2.3. Revision history

YANG packages do not contain a revision history. This is because packages may have many revisions and a long revision history would bloat the package definition. By recursively examining the 'previous-version' leaf of a package definition, a full revision history (including where non-backwards-compatible changes have occurred) can be dynamically constructed, if all package versions are available.

5.3. Package conformance

YANG packages allows for conformance to be checked at a package level rather than requiring a client to download all modules, revisions, and deviations from the server to ensure that the datastore schema used by the server is compatible with the client.

YANG package conformance is analogous to how YANG [RFC7950] requires that servers either implement a module faithfully, or otherwise use deviations to indicate areas of non-conformance.

For a top level package representing a datastore schema, servers MUST implement the package definition faithfully, including all mandatory features.

Package definitions MAY modify the schema for directly or hierarchically included packages through the use of different module revisions or module deviations. If the schema of any included package is modified in a non-backwards-compatible way then it MUST be indicated by setting the 'nbc-modified' leaf to true.

5.3.1. Use of YANG semantic versioning

Using the YANG semantic versioning scheme for package version numbers and module revision labels can help with conformance. In the general case, clients should be able to determine the nature of changes between two package versions by comparing the version number.

This usually means that a client does not have to be restricted to working only with servers that advertise exactly the same version of a package in YANG library. Instead, reasonable clients should be able to interoperate with any server that supports a package version that is backwards compatible to version that the client is designed for, assuming that the client is designed to ignore operational values for unknown data nodes.

For example, a client coded to support 'foo' package at version 1.0.0 should interoperate with a server implementing 'foo' package at

version 1.3.5, because the YANG semantic versioning rules require that package version 1.3.5 is backwards compatible to version 1.0.0.

This also has a relevance on servers that are capable of supporting version selection because they need not support every version of a YANG package to ensure good client compatibility. Choosing suitable minor versions within each major version number should generally be sufficient, particular if they can avoid NBC patch level changes (i.e. 'M' labeled versions).

5.3.2. Package checksums

Each YANG package definition may have a checksum associated with it to allow a client to validate that the package definition of the server matches the expected package definition without downloading the full package definition from the server.

The checksum for a package is calculated using the SHA-256 hash (XXX, reference) of the full file contents of the YANG package instance data file. This means that the checksum includes all whitespace and formatting, encoding, and all meta-data fields associated with the package and the instance data document).

The checksum for a module is calculated using the SHA-256 hash of the YANG module file definition. This means that the checksum includes all whitespace, formatting, and comments within the YANG module.

Packages that are locally scoped to a server may not have an offline instance data document available, and hence MAY not have a checksum.

The package definition allows URLs and checksums to be specified for all included packages, modules and submodules within the package definition. Checksums SHOULD be included in package definitions to validate the full integrity of the package.

On a server, package checksums SHOULD also be provided for the top level packages associated with the datastore schema.

5.4. Schema referential completeness

A YANG package may represent a schema that is 'referentially complete', or 'referentially incomplete', indicated in the package definition by the 'complete' flag.

If all import statements in all YANG modules included in the package (either directly, or through included packages) can be resolved to a module revision defined with the YANG package definition, then the package is classified as referentially complete. Conversely, if one

or more import statements cannot be resolved to a module specified as part of the package definition, then the package is classified as referentially incomplete.

A package that represents the exact contents of a datastore schema MUST always be referentially complete.

Referentially incomplete packages can be used, along with locally scoped packages, to represent an update to a device's datastore schema as part of an optional software hot fix. E.g., the base software is made available as a complete globally scoped package. The hot fix is made available as an incomplete globally scoped package. A device's datastore schema can define a local package that implements the base software package updated with the hot fix package.

Referentially incomplete packages could also be used to group sets of logically related modules together, but without requiring a fixed dependency on all imported 'types' modules (e.g., iana-if-types.yang), instead leaving the choice of specific revisions of 'types' modules to be resolved when the package definition is used.

5.5. Package name scoping and uniqueness

YANG package names can be globally unique, or locally scoped to a particular server or device.

5.5.1. Globally scoped packages

The name given to a package MUST be globally unique, and it MUST include an appropriate organization prefix in the name, equivalent to YANG module naming conventions.

Ideally a YANG instance data document defining a particular package version would be publicly available at one or more URLs.

5.5.2. Server scoped packages

Package definitions may be scoped to a particular server by setting the 'is-local' leaf to true in the package definition.

Locally scoped packages MAY have a package name that is not globally unique.

Locally scoped packages MAY have a definition that is not available offline from the server in a YANG instance data document.

5.6. Submodules packages considerations

As defined in [RFC7950] and [I-D.verdt-netmod-yang-semver], YANG conformance and versioning is specified in terms of particular revisions of YANG modules rather than for individual submodules.

However, YANG package definitions also include the list of submodules included by a module, primarily to provide a location of where the submodule definition can be obtained from, allowing a YANG schema to be fully constructed from a YANG package instance-data file definition.

5.7. Package tags

[I-D.ietf-netmod-module-tags] defines YANG module tags as a mechanism to annotate a module definition with additional metadata. Tags MAY also be associated to a package definition via the 'tags' leaf-list. The tags use the same registry and definitions used by YANG module tags.

6. YANG Packages instance data

YANG packages SHOULD be defined as YANG instance data documents [I-D.ietf-netmod-yang-instance-file-format] using the YANG schema below to define the package data itself.

The format of the YANG package instance file MUST follow the following rules:

The file SHOULD be encoded in JSON.

The name of the file SHOULD follow the format "<package-name>@<version>.json".

The package name MUST be specified in both the instance-data-set 'name' and package 'name' leafs.

The 'description' field of the instance-data-set SHOULD be "YANG package definition".

The 'timestamp', "organization', 'contact' fields are defined in both the instance-data-set metadata and the YANG package metadata. Package definitions SHOULD only define these fields as part of the package definition. If any of these fields are populated in the instance-data-set metadata then they MUST contain the same value as the corresponding leaves in the package definition.

The 'revision' list in the instance data document SHOULD NOT be used, since versioning is handled by the package definition.

The instance data document for each version of a YANG package SHOULD be made available at one of more locations accessible via URLs. If one of the listed locations defines a definitive reference implementation for the package definition then it MUST be listed as the first entry in the list.

The "ietf-yang-package" YANG module has the following structure:

```
module: ietf-yang-package

  structure package:
    +-- name                   pkg-identifier
    +-- version                rev:revision-label
    +-- timestamp?             yang:date-and-time
    +-- organization?          string
    +-- contact?               string
    +-- description?           string
    +-- reference?             string
    +-- location*              inet:uri
    +-- complete?              boolean
    +-- local?                 boolean
    +-- previous-version?      rev:revision-label
    +-- nbc-changes?           boolean
    +-- tag*                   tags:tag
    +-- mandatory-feature*     scoped-feature
    +-- included-package* [name version]
    |  +-- name                 pkg-identifier
    |  +-- version              rev:revision-label
    |  +-- replaces-version*    rev:revision-label
    |  +-- nbc-modified?        boolean
    |  +-- location*            inet:uri
    |  +-- checksum?            pkg-types:sha-256-hash
    +-- module* [name]
    |  +-- name                  yang:yang-identifier
    |  +-- revision?             rev:revision-date-or-label
    |  +-- replaces-revision*    rev:revision-date-or-label
    |  +-- namespace?            inet:uri
    |  +-- location*             inet:uri
    |  +-- checksum?             pkg-types:sha-256-hash
    |  +-- submodule* [name]
    |     +-- name         yang:yang-identifier
    |     +-- revision     rev:revision-identifier
    |     +-- location*    inet:uri
    |     +-- checksum?    pkg-types:sha-256-hash
```

```
      +-- import-only-module* [name revision]
         +-- name                 yang:yang-identifier
         +-- revision             rev:revision-date-or-label
         +-- replaces-revision*   rev:revision-date-or-label
         +-- namespace?           inet:uri
         +-- location*            inet:uri
         +-- checksum?            pkg-types:sha-256-hash
         +-- submodule* [name]
            +-- name        yang:yang-identifier
            +-- revision    rev:revision-identifier
            +-- location*   inet:uri
            +-- checksum?   pkg-types:sha-256-hash
```

7. YANG Packages additions to YANG library

7.1. Package List

The main addition is a top level 'yang-library/package' list that lists all versions of all packages known to the server. Each package defines a potentially incomplete YANG schema, built from included packages and module-sets. The use of module-sets allows the module definitions to be shared with the existing YANG library schema definitions. The existing rule of RFC 7995bis related to combining modules-sets also applies here, i.e. The combined set of modules defined by the module-sets MUST NOT contain modules implemented at different revisions. I.e. the module-sets leaf-list is directly equivalent to the explicit module and import-only-module lists in the instance data YANG package definition.

The 'yang-library/package' list MAY include multiple versions of a particular package. E.g. if the server is capable of allowing clients to select which package versions should be used by the server.

7.2. Binding from schema to package

The second augmentation is to allow a server to optionally indicate that a schema definition directly relates to a package. Since YANG packages are available offline, it may be sufficient for a client to only check that a compatible version of the YANG package is being implemented by the server without fetching and comparing the full module list.

If a server indicates that its schema maps to a particular package then it MUST support all features listed in the mandatory-feature list defined as part of that package, and it MUST NOT have any non-backwards-compatible deviations to the modules defined by the

package. A server MAY implement features not specified in the package's mandatory-feature list.

If a server cannot faithfully implement a package then it can define a new package to accurately report what it does implement. The new package can include the original package as an included package, and the new package can define additional modules containing deviations to the modules in the original package, allowing the new package to accurately describe the server behavior. There is no specific mechanism provided to indicate that a mandatory-feature is not supported on a server, but deviations MAY be used to disable functionality predicated by an if-feature statement.

7.3. Tree diagram

The "ietf-yang-library-packages" YANG module has the following structure:

```
   module: ietf-yl-packages
     augment /yanglib:yang-library:
       +--ro package* [name version]
          +--ro name                 pkg-identifier
          +--ro version              rev:revision-label
          +--ro timestamp?           yang:date-and-time
          +--ro organization?        string
          +--ro contact?             string
          +--ro description?         string
          +--ro reference?           string
          +--ro location*            inet:uri
          +--ro complete?            boolean
          +--ro local?               boolean
          +--ro previous-version?    rev:revision-label
          +--ro nbc-changes?         boolean
          +--ro tag*                 tags:tag
          +--ro mandatory-feature*   scoped-feature
          +--ro included-package* [name version]
          |  +--ro name                pkg-identifier
          |  +--ro version             rev:revision-label
          |  +--ro replaces-version*   rev:revision-label
          |  +--ro nbc-modified?       boolean
          |  +--ro location*           inet:uri
          |  +--ro checksum?           pkg-types:sha-256-hash
          +--ro module-set*
          |       -> /yanglib:yang-library/module-set/name
          +--ro checksum?            pkg-types:sha-256-hash
     augment /yanglib:yang-library/yanglib:schema:
       +--ro package
          +--ro name?
          |       -> /yanglib:yang-library/package/name
          +--ro version?                      leafref
          +--ro checksum?                     pkg-types:sha-256-hash
          +--ro supported-optional-feature*   pkg-types:scoped-feature
     augment /yanglib:yang-library/yanglib:module-set/yanglib:module:
       +--ro replaces-revision*   rev:revision-date-or-label
       +--ro checksum?            pkg-types:sha-256-hash
     augment /yanglib:yang-library/yanglib:module-set/yanglib:module
               /yanglib:submodule:
       +--ro checksum?   pkg-types:sha-256-hash
     augment /yanglib:yang-library/yanglib:module-set
               /yanglib:import-only-module:
       +--ro replaces-revision*   rev:revision-date-or-label
       +--ro checksum?            pkg-types:sha-256-hash
     augment /yanglib:yang-library/yanglib:module-set
               /yanglib:import-only-module/yanglib:submodule:
       +--ro checksum?   pkg-types:sha-256-hash
```

8. YANG Packages Groupings

Groupings for YANG packages related constructs are provided in a 'types' module for use by the instance-data and YANG library constructs described previously. They are also available to be used by other modules that have a need for YANG packages information.

The "ietf-yang-package-types" YANG module has the following structure:

```
module: ietf-yang-package-types

  grouping yang-pkg-identification-leafs
    +-- name       pkg-identifier
    +-- version    rev:revision-label
  grouping yang-pkg-common-leafs
    +-- timestamp?           yang:date-and-time
    +-- organization?        string
    +-- contact?             string
    +-- description?         string
    +-- reference?           string
    +-- location*            inet:uri
    +-- complete?            boolean
    +-- local?               boolean
    +-- previous-version?    rev:revision-label
    +-- nbc-changes?         boolean
    +-- tag*                 tags:tag
    +-- mandatory-feature*   scoped-feature
    +-- included-package* [name version]
       +-- name                pkg-identifier
       +-- version             rev:revision-label
       +-- replaces-version*   rev:revision-label
       +-- nbc-modified?       boolean
       +-- location*           inet:uri
       +-- checksum?           pkg-types:sha-256-hash
```

9. YANG packages as schema for YANG instance data document

YANG package definitions can be used as the schema definition for YANG instance data documents. When using a package schema, the name of the package MUST be specified, a package checksum and/or URL to the package definition MAY also be provided.

The "ietf-yang-inst-data-pkg" YANG module has the following structure:

```
module: ietf-yang-inst-data-pkg

  augment-structure /yid:instance-data-set/yid:content-schema-spec:
    +--:(pkg-schema)
      +-- pkg-schema
      +-- package       pkg-types:pkg-identifier
      +-- location*     inet:uri
      +-- checksum?     pkg-types:sha-256-hash
```

10. YANG Modules

The YANG module definitions for the modules described in the previous sections.

```
<CODE BEGINS> file "ietf-yang-package-types@2019-09-11.yang"
module ietf-yang-package-types {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-package-types";
  prefix "pkg-types";

  import ietf-yang-revisions {
    prefix rev;
    reference "XXXX: Updated YANG Module Revision Handling";
  }

  import ietf-yang-types {
    prefix yang;
    rev:revision-or-derived 2013-07-15;
    reference "RFC 6991: Common YANG Data Types.";
  }

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types.";
  }

  import ietf-module-tags {
    prefix tags;
    reference "RFC XXX: YANG Module Tags.";
  }

  organization
```

```
       "IETF NETMOD (Network Modeling) Working Group";

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

        Author:   Rob Wilton
                  <mailto:rwilton@cisco.com>";

     description
       "This module provides type and grouping definitions for YANG
        packages.

        Copyright (c) 2019 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; see
        the RFC itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
        NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
        'MAY', and 'OPTIONAL' in this document are to be interpreted as
        described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
        they appear in all capitals, as shown here.";

     // RFC Ed.: update the date below with the date of RFC publication
     // and remove this note.
     // RFC Ed.: replace XXXX with actual RFC number and remove this
     // note.
     revision 2019-09-11 {
       rev:revision-label 0.1.0;
       description
         "Initial revision";
       reference
         "RFC XXXX: YANG Packages";
     }


     /*
      * Typedefs
      */
```

```
typedef pkg-identifier {
  type yang:yang-identifier;
  description
    "Package identifiers are typed as YANG identifiers.";
}

typedef scoped-feature {
  type string {
    pattern '[a-zA-Z_][a-zA-Z0-9\-_.]*:[a-zA-Z_][a-zA-Z0-9\-_.]*';
  }
  description
    "Represents a feature name scoped to a particular module,
     identified as the '<module-name>:<feature-name>', where both
     <module-name> and <feature-name> are YANG identifier strings,
     as defiend by Section 12 or RFC 6020.";
  reference
    "RFC XXXX, YANG Packages.";
}

typedef sha-256-hash {
  type string {
    length "64";
    pattern "[0-9a-fA-F]*";
  }
  description
    "A SHA-256 hash represented as a hexadecimal string.

     Used as the checksum for modules, submodules and packages in a
     YANG package definition.

     For modules and submodules the SHA-256 hash is calculated on
     the contents of the YANG file defining the module/submodule.

     For packages the SHA-256 hash is calculated on the file
     containing the YANG instance data document holding the package
     definition";
}


/*
 * Groupings
 */

grouping yang-pkg-identification-leafs {
  description
    "Parameters for identifying a specific version of a YANG
     package";
```

```
      leaf name {
        type pkg-identifier;
        mandatory true;
        description
          "The YANG package name.";
      }

      leaf version {
        type rev:revision-label;
        mandatory true;
        description
          "Uniquely identies a particular version of a YANG package.

           Follows the definition for revision labels defined in
           draft-verdt-nemod-yang-module-versioning, section XXX";
      }
    }

    grouping yang-pkg-common-leafs {
      description
        "Defines definitions common to all YANG package definitions.";

      leaf timestamp {
        type yang:date-and-time;

        description
          "An optional timestamp for when this package was created.
           This does not need to be unique across all versions of a
           package.";
      }

      leaf organization {
        type string;

        description "Organization responsible for this package";
      }

      leaf contact {
        type string;

        description
          "Contact information for the person or organization to whom
           queries concerning this package should be sent.";
      }

      leaf description {
        type string;
```

```
        description "Provides a description of the package";
      }

      leaf reference {
        type string;

        description "Allows for a reference for the package";
      }

      leaf-list location {
        type inet:uri;
        description
          "Contains a URL that represents where an instance data file
           for this YANG package can be found.

           This leaf will only be present if there is a URL
           available for retrieval of the schema for this entry.

           If multiple locations are provided, then the first location
           in the leaf-list MUST be the definitive location that
           uniquely identifies this package";
      }

      leaf complete {
        type boolean;
        default true;
        description
          "Indicates whether the schema defined by this package is
           referentially complete.  I.e. all module imports can be
           resolved to a module explicitly defined in this package or
           one of the included packages.";
      }

      leaf local {
        type boolean;
        default false;
        description
          "Defines that the package definition is local to the server,
           and the name of the package MAY not be unique, and the
           package definition MAY not be available in an offline file.

           Local packages can be used when the schema for the device
           can be changed at runtime through the addition or removal of
           software packages, or hot fixes.";
      }

      leaf previous-version {
        type rev:revision-label;
```

```
        description
          "The previous package version that this version has been
           derived from.  This leaf allows a full version history graph
           to be constructed if required.";
      }

      leaf nbc-changes {
        type boolean;
        default false;
        description
          "Indicates whether the defined package version contains
           non-backwards-compatible changes relative to the package
           version defined in the 'previous-version' leaf.";
      }

      leaf-list tag {
        type tags:tag;
        description
          "Tags associated with a YANG package.  Module tags defined in
           XXX, ietf-netmod-module-tags can be used here but with the
           modification that the tag applies to the entire package
           rather than a specific module.  See the IANA 'YANG Module
           Tag Prefix' registry for reserved prefixes and the IANA
           'YANG Module IETF Tag' registry for IETF standard tags.";
      }

      leaf-list mandatory-feature {
        type scoped-feature;
        description
          "Lists features from any modules included in the package that
           MUST be supported by any server implementing the package.

           Features already specified in a 'mandatory-feature' list of
           any included package MUST also be supported by server
           implementations and do not need to be repeated in this list.

           All other features defined in modules included in the
           package are OPTIONAL to implement.

           Features are identified using <module-name>:<feature-name>";
      }

      list included-package {
        key "name version";
        description
          "An entry in this list represents a package that is included
           as part of the package definition, or an indirectly included
           package that is changed in a non backwards compatible way.
```

```
          It can be used to resolve inclusion of conflicting package
          versions by explicitly specifying which package version is
          used.

          If included packages implement different revisions or
          versions of the same module, then an explicit entry in the
          module list MUST be provided to select the specific module
          version 'implemented' by this package definition.

          If the schema for any packages that are included, either
          directly or indirectly via another package include, are
          changed in any non-backwards-compatible way then they MUST
          be explicitly listed in the included-packages list with the
          'nbc-modified' leaf set to true.

          For import-only modules, the 'replaces-revision' leaf-list
          can be used to select the specific module versions used
          by this package.";
      reference
        "XXX";

      uses yang-pkg-identification-leafs;

      leaf-list replaces-version {
        type rev:revision-label;
        description
          "Gives the version of an included package version that
           is replaced by this included package revision.";
      }

      leaf nbc-modified {
        type boolean;
        default false;
        description
          "Set to true if any data nodes in this package are modified
           in a non backwards compatible way, either through the use
           of deviations, or because one of the modules has been
           replaced by an incompatible revision.  This could also
           occur if a module's revision was replaced by an earlier
           revision that had the effect of removing some data
           nodes.";
      }

      leaf-list location {
        type inet:uri;
        description
          "Contains a URL that represents where an instance data file
           for this YANG package can be found.
```

```
              This leaf will only be present if there is a URL available
              for retrieval of the schema for this entry.

              If multiple locations are provided, then the first
              location in the leaf-list MUST be the definitive location
              that uniquely identifies this package";
          }

          leaf checksum {
            type pkg-types:sha-256-hash;
            description
              "The SHA-256 hash calculated on the textual package
               definition, represented as a hexadecimal string.";
          }
        }
      }
    }
    <CODE ENDS>



    <CODE BEGINS> file "ietf-yang-package@2019-09-11.yang"
    module ietf-yang-package {
      yang-version 1.1;
      namespace "urn:ietf:params:xml:ns:yang:ietf-yang-package";
      prefix pkg;

      import ietf-yang-revisions {
        prefix rev;
        reference "XXXX: Updated YANG Module Revision Handling";
      }

      import ietf-yang-package-types {
        prefix pkg-types;
        rev:revision-or-derived 0.1.0;
        reference "RFC XXX: YANG Schema Versioning.";
      }

      import ietf-yang-structure-ext {
        prefix sx;
        reference "RFC XXX: YANG Data Structure Extensions.";
      }

      import ietf-yang-types {
        prefix yang;
        rev:revision-or-derived 2013-07-15;
        reference "RFC 6991: Common YANG Data Types.";
      }
```

```
  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types.";
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";

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

     Author:   Rob Wilton
               <mailto:rwilton@cisco.com>";

  description
    "This module provides a definition of a YANG package, which is
     used as the schema for an YANG instance data document specifying
     a YANG package.

     Copyright (c) 2019 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; see
     the RFC itself for full legal notices.

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
     NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
     'MAY', and 'OPTIONAL' in this document are to be interpreted as
     described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
     they appear in all capitals, as shown here.";

  // RFC Ed.: update the date below with the date of RFC publication
  // and remove this note.
  // RFC Ed.: replace XXXX with actual RFC number and remove this
  // note.
  revision 2019-09-11 {
    rev:revision-label 0.1.0;
    description
      "Initial revision";
    reference
```

```
        "RFC XXXX: YANG Packages";
    }


    /*
     * Top-level structure
     */

    sx:structure package {
      description
        "Defines a YANG package.

         Intended to be used to specify YANG package within an instance
         data document.";

      uses pkg-types:yang-pkg-identification-leafs;
      uses pkg-types:yang-pkg-common-leafs;

      list module {
        key "name";
        description
          "An entry in this list represents a module that must be
           implemented by a server implementing this package, as per
           RFC 7950 section 5.6.5, with a particular set of supported
           features and deviations.

           A entry in this list overrides any module revision
           'implemented' by an included package.  Any replaced module
           revision SHOULD also be listed in the 'replaces-revision'
           list.";
        reference
          "RFC 7950: The YANG 1.1 Data Modeling Language.";

        leaf name {
          type yang:yang-identifier;
          mandatory true;
          description
            "The YANG module name.";
        }

        leaf revision {
          type rev:revision-date-or-label;
          description
            "The YANG module revision date or revision-label.

             If no revision statement is present in the YANG module,
             this leaf is not instantiated.";
        }
```

```
        leaf-list replaces-revision {
          type rev:revision-date-or-label;
          description
            "Gives the revision of an module (implemented or
             import-only) defined in an included package that is
             replaced by this implemented module revision.";
        }

        leaf namespace {
          type inet:uri;
          description
            "The XML namespace identifier for this module.";
        }

        leaf-list location {
          type inet:uri;
          description
            "Contains a URL that represents the YANG schema resource
             for this module.

             This leaf will only be present if there is a URL available
             for retrieval of the schema for this entry.";
        }

        leaf checksum {
          type pkg-types:sha-256-hash;
          description
            "The SHA-256 hash calculated on the textual module
             definition, represented as a hexadecimal string.";
        }

        list submodule {
          key "name";
          description
            "Each entry represents one submodule within the
             parent module.";

          leaf name {
            type yang:yang-identifier;
            description
              "The YANG submodule name.";
          }

          leaf revision {
            type rev:revision-identifier;
            mandatory true;
            description
              "The YANG submodule revision date.  If the parent module
```

```
            include statement for this submodule includes a revision
            date then it MUST match this leaf's value.";
        }

        leaf-list location {
          type inet:uri;
          description
            "Contains a URL that represents the YANG schema resource
             for this submodule.

             This leaf will only be present if there is a URL
             available for retrieval of the schema for this entry.";
        }

        leaf checksum {
          type pkg-types:sha-256-hash;
          description
            "The SHA-256 hash calculated on the textual submodule
             definition, represented as a hexadecimal string.";
        }
      }
    }

    list import-only-module {
      key "name revision";
      description
        "An entry in this list indicates that the server imports
         reusable definitions from the specified revision of the
         module, but does not implement any protocol accessible
         objects from this revision.

         Multiple entries for the same module name MAY exist.  This
         can occur if multiple modules import the same module, but
         specify different revision-dates in the import statements.";

      leaf name {
        type yang:yang-identifier;
        description
          "The YANG module name.";
      }

      leaf revision {
        type rev:revision-date-or-label;
        description
          "The YANG module revision date or revision-label.

           If no revision statement is present in the YANG module,
           this leaf is not instantiated.";
```

```
        }

        leaf-list replaces-revision {
          type rev:revision-date-or-label;
          description
            "Gives the revision of an import-only-module defined in an
             included package that is replaced by this
             import-only-module revision.";
        }

        leaf namespace {
          type inet:uri;
          description
            "The XML namespace identifier for this module.";
        }

        leaf-list location {
          type inet:uri;
          description
            "Contains a URL that represents the YANG schema resource
             for this module.

             This leaf will only be present if there is a URL available
             for retrieval of the schema for this entry.";
        }

        leaf checksum {
          type pkg-types:sha-256-hash;
          description
            "The SHA-256 hash calculated on the textual submodule
             definition, represented as a hexadecimal string.";
        }

        list submodule {
          key "name";
          description
            "Each entry represents one submodule within the
             parent module.";

          leaf name {
            type yang:yang-identifier;
            description
              "The YANG submodule name.";
          }

          leaf revision {
            type rev:revision-identifier;
            mandatory true;
```

```
            description
              "The YANG submodule revision date.  If the parent module
               include statement for this submodule includes a revision
               date then it MUST match this leaf's value.";
          }

          leaf-list location {
            type inet:uri;
            description
              "Contains a URL that represents the YANG schema resource
               for this submodule.

               This leaf will only be present if there is a URL
               available for retrieval of the schema for this entry.";
          }

          leaf checksum {
            type pkg-types:sha-256-hash;
            description
              "The SHA-256 hash calculated on the textual submodule
               definition, represented as a hexadecimal string.";
          }
        }
      }
    }
  }
  <CODE ENDS>



  <CODE BEGINS> file "ietf-yl-packages@2019-09-11.yang"
  module ietf-yl-packages {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-yl-packages";
    prefix yl-pkg;

    import ietf-yang-revisions {
      prefix rev;
      reference "XXXX: Updated YANG Module Revision Handling";
    }

    import ietf-yang-package-types {
      prefix pkg-types;
      reference "RFC XXX: YANG Packages.";
    }

    import ietf-yang-library {
      prefix yanglib;
```

```
      reference "RFC 7895bis: YANG Library";
    }

    organization
      "IETF NETMOD (Network Modeling) Working Group";

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

       Author:   Rob Wilton
                 <mailto:rwilton@cisco.com>";

    description
      "This module provides defined augmentations to YANG library to
       allow a server to report YANG package information.

       Copyright (c) 2018 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; see
       the RFC itself for full legal notices.

       The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
       NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
       'MAY', and 'OPTIONAL' in this document are to be interpreted as
       described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
       they appear in all capitals, as shown here.";


    // RFC Ed.: update the date below with the date of RFC publication
    // and remove this note.
    // RFC Ed.: replace XXXX with actual RFC number and remove this
    // note.
    revision 2019-09-11 {
      rev:revision-label 0.1.0;
      description
        "Initial revision";
      reference
        "RFC XXXX: YANG Packages";
    }
```

```
     /*
      * Augmentations
      */

     augment "/yanglib:yang-library" {
       description "Add YANG package definitions into YANG library";

       list package {
         key "name version";
         config "false";

         description
           "Defines the package available on this server.";

         uses pkg-types:yang-pkg-identification-leafs;
         uses pkg-types:yang-pkg-common-leafs;

         leaf-list module-set {
           type leafref {
             path "/yanglib:yang-library/yanglib:module-set/" +
                  "yanglib:name";
           }
           description
             "Describes any modules in addition to, and replacing, and
              modules defined in the included packages.

              If a non import-only module appears in multiple module
              sets, then the module revision and the associated features
              and deviations MUST be identical.";
         }

         leaf checksum {
           type pkg-types:sha-256-hash;
           description
             "The SHA-256 hash calculated on the textual package
              definition, represented as a hexadecimal string.";
         }
       }
     }

     augment "/yanglib:yang-library/yanglib:schema" {
       description
         "Allow datastore schema to be related to a YANG package";

       container package {
          leaf name {
            type leafref {
              path "/yanglib:yang-library/package/name";
```

```
        }
        description
          "The name of the package this schema relates to.

           The referenced package MUST represent a referentially
           complete schema";
      }

      leaf version {
        type leafref {
          path '/yanglib:yang-library/'
            + 'package[name = current()/../name]/version';
        }

        description
          "The version number of the package this schema relates
           to.";
      }

      leaf checksum {
        type pkg-types:sha-256-hash;
        description
          "The checksum of the package this schema relates to,
           calculated on the 'YANG instance data file' package
           definition.

           This leaf MAY be omitted if the referenced package is
           locally scoped without an associated checksum.";
      }

      leaf-list supported-optional-feature {
        type pkg-types:scoped-feature;
        description
          "Lists all optional module features that are also
           supported by the server when implementing the package.

           This list SHOULD exclude any features in the
           'mandatory-feature' list for the package, or any included
           package.

           The full set of features supported by the server for this
           schema is the union of this list and all
           'mandatory-feature' lists for the package and all
           included packages.  This is equivalent to the information
           provided via the 'feature' leaf list in YANG library.

           Features are identified using
           '<module-name>:<feature-name>'";
```

```
          }

        description
          "Describes which package the schema directly relates to, if
           any.";
      }
    }

    augment "/yanglib:yang-library/yanglib:module-set/yanglib:module" {

      description
        "Add 'replaced-revision' and 'checksum' to implemented module
         definitions.";

      leaf-list replaces-revision {
        type rev:revision-date-or-label;
        description
          "Gives the revision of an module (implemented or import-only)
           defined in an included package that is replaced by this
           implemented module revision.

           Only used for YANG package definitions";
      }

      leaf checksum {
        type pkg-types:sha-256-hash;
        description
          "The SHA-256 hash calculated on the textual module
           definition, represented as a hexadecimal string.";
      }
    }

    augment
      "/yanglib:yang-library/yanglib:module-set/" +
      "yanglib:module/yanglib:submodule" {

      description
        "Add 'checksum' to implemented modules' submodule
         definitions.";

      leaf checksum {
        type pkg-types:sha-256-hash;
        description
          "The SHA-256 hash calculated on the textual submodule
           definition, represented as a hexadecimal string.";
      }
    }
```

```
    augment "/yanglib:yang-library/yanglib:module-set/" +
            "yanglib:import-only-module" {

      description
        "Add 'replaces-revision' and 'checksum' to import-only-module
         definitions";

      leaf-list replaces-revision {
        type rev:revision-date-or-label;
        description
          "Gives the revision of an import-only-module defined in an
           imported package that is replaced by this import-only-module
           revision.

           Only used for YANG package definitions";
      }

      leaf checksum {
        type pkg-types:sha-256-hash;
        description
          "The SHA-256 hash calculated on the textual module
           definition, represented as a hexadecimal string.";
      }
    }

    augment "/yanglib:yang-library/yanglib:module-set/" +
            "yanglib:import-only-module/yanglib:submodule" {

      description
        "Add 'checksum' to import-only-modules' submodule
         definitions.";

      leaf checksum {
        type pkg-types:sha-256-hash;
        description
          "The SHA-256 hash calculated on the textual submodule
           definition, represented as a hexadecimal string.";
      }
    }
  }
  <CODE ENDS>



  <CODE BEGINS> file "ietf-yang-inst-data-pkg@2019-09-11.yang"
  module ietf-yang-inst-data-pkg {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-yang-inst-data-pkg";
```

```
  prefix yid-pkg;

  import ietf-yang-revisions {
    prefix rev;
    reference "XXXX: Updated YANG Module Revision Handling";
  }

  import ietf-yang-package-types {
    prefix pkg-types;
    rev:revision-or-derived 0.1.0;
    reference "RFC XXX: YANG Schema Versioning.";
  }

  import ietf-yang-structure-ext {
    prefix sx;
    reference "RFC XXX: YANG Data Structure Extensions.";
  }

  import ietf-yang-instance-data {
    prefix yid;
    reference "RFC XXX: YANG Instance Data File Format.";
  }

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types.";
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";

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

     Author:   Rob Wilton
               <mailto:rwilton@cisco.com>";

  description
    "The module augments ietf-yang-instance-data to allow package
     definitions to be used to define schema in YANG instance data
     documents.

     Copyright (c) 2019 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; see
        the RFC itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
        NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
        'MAY', and 'OPTIONAL' in this document are to be interpreted as
        described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
        they appear in all capitals, as shown here.";

    // RFC Ed.: update the date below with the date of RFC publication
    // and remove this note.
    // RFC Ed.: replace XXXX with actual RFC number and remove this
    // note.
    revision 2019-09-11 {
      rev:revision-label 0.1.0;
      description
        "Initial revision";
      reference
        "RFC XXXX: YANG Packages";
    }

    /*
     * Augmentations
     */

    sx:augment-structure
      "/yid:instance-data-set/yid:content-schema-spec" {
      description
        "Add package reference to instance data set schema
         specification";
      case pkg-schema {
        container pkg-schema {
          leaf pkg-schema {
            type pkg-types:pkg-identifier;
            mandatory true;
            description
              "The package definition that defines the schema for this
               file.";
          }
          leaf checksum {
            type pkg-types:sha-256-hash;
            description
              "The SHA-256 hash of the package, calculated on
```

```
                the textual package definition, represented as a
                hexadecimal string.";
            }
            leaf-list location {
              type inet:uri;
              description
                "Contains a URL that represents where an instance data
                 file for this YANG package can be found.

                 This leaf will only be present if there is a URL
                 available for retrieval of the schema for this entry.

                 If multiple locations are provided, then the first
                 location in the leaf-list MUST be the definitive
                 location that uniquely identifies this package";
            }
          }
        }
      }
    }
  }
  <CODE ENDS>
```

11. Security Considerations

The YANG modules specified in this document defines a schema for data that is accessed by network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

Similarly to YANG library [I-D.ietf-netconf-rfc7895bis], some of the readable data nodes in these YANG modules may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes.

One additional key different to YANG library, is that the 'ietf-yang-package' YANG module defines a schema to allow YANG packages to be defined in YANG instance data documents, that are outside the security controls of the network management protocols. Hence, it is

important to also consider controlling access to these package instance data documents to restrict access to sensitive information. SHA-256 checksums are used to ensure the integrity of YANG package definitions, imported modules, and sub-modules.

As per the YANG library security considerations, the module, revision and version information in YANG packages may help an attacker identify the server capabilities and server implementations with known bugs since the set of YANG modules supported by a server may reveal the kind of device and the manufacturer of the device. Server vulnerabilities may be specific to particular modules, module revisions, module features, or even module deviations. For example, if a particular operation on a particular data node is known to cause a server to crash or significantly degrade device performance, then the YANG packages information will help an attacker identify server implementations with such a defect, in order to launch a denial-of-service attack on the device.

12. IANA Considerations

It is expected that a central registry of standard YANG package definitions is required to support this solution.

It is unclear whether an IANA registry is also required to manage specific package versions. It is highly desirable to have a specific canonical location, under IETF control, where the definitive YANG package versions can be obtained from.

This document requests IANA to registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registrations are requested.

```
URI: urn:ietf:params:xml:ns:yang:ietf-yang-package-types.yang
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-yang-package.yang
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-yl-packages.yang
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.
```

This document requests that the following YANG modules are added in the "YANG Module Names" registry [RFC6020]:

```
Name: ietf-yang-package-types.yang
```

```
Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-package-
types.yang
Prefix: pkg-types
Reference: RFC XXXX

Name: ietf-yang-package.yang
Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-package.yang
Prefix: pkg
Reference: RFC XXXX

Name: ietf-yl-packages.yang
Namespace: urn:ietf:params:xml:ns:yang:ietf-yl-packages.yang
Prefix: yl-pkg
Reference: RFC XXXX
```

13. Open Questions/Issues

All issues, along with the draft text, are currently being tracked at https://github.com/rgwilton/YANG-Packages-Draft/issues/

14. Acknowledgements

Feedback helping shape this document has kindly been provided by Andy Bierman, Joe Clarke, James Cumming, Mahesh Jethanandani, Balazs Lengyel, Ladislav Lhotka, Jason Sterne, and Reshad Rahman.

15. References

15.1. Normative References

[I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-07 (work in progress), October 2018.

[I-D.ietf-netmod-module-tags]
Hopps, C., Berger, L., and D. Bogdanovic, "YANG Module Tags", draft-ietf-netmod-module-tags-09 (work in progress), September 2019.

[I-D.ietf-netmod-yang-data-ext]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Data Structure Extensions", draft-ietf-netmod-yang-data-ext-04 (work in progress), July 2019.

[I-D.ietf-netmod-yang-instance-file-format]
Lengyel, B. and B. Claise, "YANG Instance Data File Format", draft-ietf-netmod-yang-instance-file-format-04 (work in progress), August 2019.

[I-D.verdt-netmod-yang-module-versioning]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "Updated YANG Module Revision Handling", draft-verdt-netmod-yang-module-versioning-01 (work in progress), October 2019.

[I-D.verdt-netmod-yang-semver]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "YANG Semantic Versioning", draft-verdt-netmod-yang-semver-01 (work in progress), October 2019.

[I-D.verdt-netmod-yang-solutions]
Wilton, R., "YANG Versioning Solution Overview", draft-verdt-netmod-yang-solutions-01 (work in progress), July 2019.

[I-D.verdt-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", draft-verdt-netmod-yang-versioning-reqs-02 (work in progress), November 2018.

[I-D.wilton-netmod-yang-ver-selection]
Wilton, R. and R. Rahman, "YANG Schema Version Selection", draft-wilton-netmod-yang-ver-selection-00 (work in progress), March 2019.

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc6242>.

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

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

15.2. Informative References

[I-D.bierman-netmod-yang-package] Bierman, A., "The YANG Package Statement", draft-bierman-netmod-yang-package-00 (work in progress), July 2015.

[I-D.ietf-netmod-artwork-folding]
Watsen, K., Farrel, A., and Q. WU, "Handling Long Lines in Inclusions in Internet-Drafts and RFCs", draft-ietf-netmod-artwork-folding-10 (work in progress), September 2019.

[openconfigsemver]
"Semantic Versioning for OpenConfig Models", <http://www.openconfig.net/docs/semver/>.

[RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <https://www.rfc-editor.org/info/rfc8199>.

Appendix A. Tree output for ietf-yang-library with package augmentations

Complete tree output for ietf-yang-library with package augmentations.

```
module: ietf-yang-library
    +--ro yang-library
    |  +--ro module-set* [name]
    |  |  +--ro name                    string
    |  |  +--ro module* [name]
    |  |  |  +--ro name          yang:yang-identifier
    |  |  |  +--ro revision?     revision-identifier
    |  |  |  +--ro namespace     inet:uri
    |  |  |  +--ro location*     inet:uri
    |  |  |  +--ro submodule* [name]
    |  |  |  |  +--ro name           yang:yang-identifier
    |  |  |  |  +--ro revision?      revision-identifier
    |  |  |  |  +--ro location*      inet:uri
    |  |  |  |  +--ro yl-pkg:checksum?  pkg-types:sha-256-hash
    |  |  |  +--ro feature*      yang:yang-identifier
    |  |  |  +--ro deviation*    -> ../../module/name
    |  |  |  +--ro yl-pkg:replaces-revision*
    |  |  |  |       yanglib:revision-identifier
    |  |  |  +--ro yl-pkg:checksum?  pkg-types:sha-256-hash
    |  |  +--ro import-only-module* [name revision]
    |  |     +--ro name                      yang:yang-identifier
    |  |     +--ro revision                  union
    |  |     +--ro namespace                 inet:uri
    |  |     +--ro location*                 inet:uri
    |  |     +--ro submodule* [name]
    |  |     |  +--ro name           yang:yang-identifier
    |  |     |  +--ro revision?      revision-identifier
```

```
|  |  |     |  +--ro location*      inet:uri
|  |  |     |  +--ro pkg:checksum?  pkg-types:sha-256-hash
|  |  |     +--ro yl-pkg:replaces-revision*
|  |  |     |       yanglib:revision-identifier
|  |  |     +--ro yl-pkg:checksum?  pkg-types:sha-256-hash
|  |  +--ro schema* [name]
|  |  |  +--ro name           string
|  |  |  +--ro module-set*    -> ../../module-set/name
|  |  |  +--ro yl-pkg:package
|  |  |     +--ro yl-pkg:name?
|  |  |     |       -> /yanglib:yang-library/package/name
|  |  |     +--ro yl-pkg:version?         leafref
|  |  |     +--ro yl-pkg:supported-feature* pkg-types:scoped-feature
|  |  +--ro datastore* [name]
|  |  |  +--ro name      ds:datastore-ref
|  |  |  +--ro schema    -> ../../schema/name
|  |  +--ro content-id     string
|  |  +--ro yl-pkg:package* [name version]
|  |     +--ro yl-pkg:name                       yang:yang-identifier
|  |     +--ro yl-pkg:version                    rev:revision-label
|  |     +--ro yl-pkg:timestamp?                 yang:date-and-time
|  |     +--ro yl-pkg:organization?              string
|  |     +--ro yl-pkg:contact?                   string
|  |     +--ro yl-pkg:description?               string
|  |     +--ro yl-pkg:reference?                 string
|  |     +--ro yl-pkg:complete?                  boolean
|  |     +--ro yl-pkg:location*                  inet:uri
|  |     +--ro yl-pkg:local?                     boolean
|  |     +--ro yl-pkg:previous-version?          yang-sem-ver
|  |     +--ro yl-pkg:nbc-changes?               boolean
|  |     +--ro yl-pkg:tag*                       tags:tag
|  |     +--ro yl-pkg:mandatory-feature*         string
|  |     +--ro yl-pkg:included-package* [name version]
|  |     |  +--ro yl-pkg:name                    yang:yang-identifier
|  |     |  +--ro yl-pkg:version                 rev:revision-label
|  |     |  +--ro yl-pkg:replaces-version*       rev:revision-label
|  |     |  +--ro yl-pkg:nbc-modified?           boolean
|  |     |  +--ro yl-pkg:location*               inet:uri
|  |     |  +--ro yl-pkg:checksum?               string
|  |     +--ro yl-pkg:module-set*
|  |     |       -> /yanglib:yang-library/module-set/name
|  |     +--ro yl-pkg:checksum?                pkg-types:sha-256-hash
|  x--ro modules-state
|     x--ro module-set-id    string
|     x--ro module* [name revision]
|        x--ro name                yang:yang-identifier
|        x--ro revision            union
|        +--ro schema?             inet:uri
```

```
            x--ro namespace           inet:uri
            x--ro feature*            yang:yang-identifier
            x--ro deviation* [name revision]
            |  x--ro name         yang:yang-identifier
            |  x--ro revision     union
            x--ro conformance-type    enumeration
            x--ro submodule* [name revision]
               x--ro name         yang:yang-identifier
               x--ro revision     union
               +--ro schema?      inet:uri

     notifications:
       +---n yang-library-update
       |  +--ro content-id    -> /yang-library/content-id
       x---n yang-library-change
          x--ro module-set-id    -> /modules-state/module-set-id
```

Appendix B. Examples

This section provides various examples of YANG packages, and as such this text is non-normative. The purpose of the examples is to only illustrate the file format of YANG packages, and how package dependencies work. It does not imply that such packages will be defined by IETF, or which modules would be included in those packages even if they were defined. For brevity, the examples exclude namespace declarations, and use a shortened URL of "tiny.cc/ietf-yang" as a replacement for "https://raw.githubusercontent.com/YangModels/yang/master/standard/ietf/RFC".

B.1. Example IETF Network Device YANG package

This section provides an instance data document example of an IETF Network Device YANG package formatted in JSON.

This example package is intended to represent the standard set of YANG modules, with import dependencies, to implement a basic network device without any dynamic routing or layer 2 services. E.g., it includes functionality such as system information, interface and basic IP configuration.

As for all YANG packages, all import dependencies are fully resolved. Because this example uses YANG modules that have been standardized before YANG semantic versioning, they modules are referenced by revision date rather than version number.

```
   <CODE BEGINS> file "example-ietf-network-device-pkg.json"
   ========= NOTE: '\' line wrapping per BCP XX (RFC XXXX) ===========

    {
     "ietf-yang-instance-data:instance-data-set": {
       "name": "example-ietf-network-device-pkg",
       "pkg-schema": {
          package: "ietf-yang-package-defn-pkg@0.1.0.json"
       },
       "description": "YANG package definition",
       "content-data": {
         "ietf-yang-package:yang-package": {
           "name": "example-ietf-network-device-pkg",
           "version": "1.1.2",
           "timestamp": "2018-12-13T17:00:00Z",
           "organization": "IETF NETMOD Working Group",
           "contact" : "WG Web:   <http://tools.ietf.org/wg/netmod/>, \
                        WG List:  <mailto:netmod@ietf.org>",
           "description": "Example IETF network device YANG package.\
              \
              This package defines a small sample set of \
              YANG modules that could represent the basic set of \
              modules that a standard network device might be expected \
              to support.",
           "reference": "XXX, draft-rwilton-netmod-yang-packages",
           "location": [ "file://example.org/yang/packages/\
                                    ietf-network-device@v1.1.2.json" ],
           "module": [
             {
               "name": "iana-crypt-hash",
               "revision": "2014-08-06",
               "location": [ "https://tiny.cc/ietf-yang/\
                                  iana-crypt-hash%402014-08-06.yang" ],
               "checksum": "fa9fde408ddec2c16bf2c6b9e4c2f80b\
                            813a2f9e48c127016f3fa96da346e02d"
             },
             {
               "name": "ietf-system",
               "revision": "2014-08-06",
               "location": [ "https://tiny.cc/ietf-yang/\
                                    ietf-system%402014-08-06.yang" ],
               "checksum": "8a692ee2521b4ffe87a88303a61a1038\
                            79ee26bff050c1b05a2027ae23205d3f"
             },
             {
               "name": "ietf-interfaces",
               "revision": "2018-02-20",
               "location": [ "https://tiny.cc/ietf-yang/\
```

```
                                ietf-interfaces%402018-02-20.yang" ],
          "checksum": "f6faea9938f0341ed48fda93dba9a69a\
                       a32ee7142c463342efec3d38f4eb3621"
        },
        {
          "name": "ietf-netconf-acm",
          "revision": "2018-02-14",
          "location": [ "https://tiny.cc/ietf-yang/\
                              ietf-netconf-acm%402018-02-14.yang" ],
          "checksum": "e03f91317f9538a89296e99df3ff0c40\
                       03cdfea70bf517407643b3ec13c1ed25"
        },
        {
          "name": "ietf-key-chain",
          "revision": "2017-06-15",
          "location": [ "https://tiny.cc/ietf-yang/\
                                   ietf-key-chain@2017-06-15.yang" ],
          "checksum": "6250705f59fc9ad786e8d74172ce90d5\
                       8deec437982cbca7922af40b3ae8107c"
        },
        {
          "name": "ietf-ip",
          "revision": "2018-02-22",
          "location": [ "https://tiny.cc/ietf-yang/\
                                        ietf-ip%402018-02-22.yang" ],
          "checksum": "b624c84a66c128ae69ab107a5179ca8e\
                       20e693fb57dbe5cb56c3db2ebb18c894"
      }
      ],
      "import-only-module": [
        {
          "name": "ietf-yang-types",
          "revision": "2013-07-15",
          "location": [ "https://tiny.cc/ietf-yang/\
                                ietf-yang-types%402013-07-15.yang" ],
          "checksum": "a04cdcc875764a76e89b7a0200c6b9d8\
                       00b10713978093acda7840c7c2907c3f"
        },
        {
          "name": "ietf-inet-types",
          "revision": "2013-07-15",
          "location": [ "https://tiny.cc/ietf-yang/\
                                ietf-inet-types%402013-07-15.yang" ],
          "checksum": "12d98b0143a5ca5095b36420f9ebc1ff\
                       a61cfd2eaa850080244cadf01b86ddf9"
        }
      ]
    }
```

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

B.2. Example IETF Basic Routing YANG package

This section provides an instance data document example of a basic IETF Routing YANG package formatted in JSON.

This example package is intended to represent the standard set of YANG modules, with import dependencies, that builds upon the example-ietf-network-device YANG package to add support for basic dynamic routing and ACLs.

As for all YANG packages, all import dependencies are fully resolved. Because this example uses YANG modules that have been standardized before YANG semantic versioning, they modules are referenced by revision date rather than version number. Locations have been excluded where they are not currently known, e.g., for YANG modules defined in IETF drafts. In a normal YANG package, locations would be expected to be provided for all YANG modules.

```
<CODE BEGINS> file "example-ietf-routing-pkg.json"
========== NOTE: '\' line wrapping per BCP XX (RFC XXXX) ==========

{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-ietf-routing-pkg",
    "module": [ "ietf-yang-package@2019-09-11.yang" ],
    "description": "YANG package definition",
    "content-data": {
      "ietf-yang-package:yang-package": {
        "name": "example-ietf-routing",
        "version": "1.3.1",
        "timestamp": "2018-12-13T17:00:00Z",
        "description": "This package defines a small sample set of \
          IETF routing YANG modules that could represent the set of \
          IETF routing functionality that a basic IP network device \
          might be expected to support.",
        "reference": "XXX, draft-rwilton-netmod-yang-packages",
        "imported-packages": [
          {
            "name": "ietf-network-device",
            "version": "1.1.2",
            "location": [ "http://example.org/yang/packages/\
```

```
                                    ietf-network-device@v1.1.2.json" ],
        "checksum": ""
      }
    ],
    "module": [
      {
        "name": "ietf-routing",
        "revision": "2018-03-13",
        "location": [ "https://tiny.cc/ietf-yang/\
                                     ietf-routing@2018-03-13.yang" ],
        "checksum": ""
      },
      {
        "name": "ietf-ipv4-unicast-routing",
        "revision": "2018-03-13",
        "location": [ "https://tiny.cc/ietf-yang/\
                     ietf-ipv4-unicast-routing@2018-03-13.yang" ],
        "checksum": ""
      },
      {
        "name": "ietf-ipv6-unicast-routing",
        "revision": "2018-03-13",
        "location": [ "https://tiny.cc/ietf-yang/\
                     ietf-ipv6-unicast-routing@2018-03-13.yang" ],
        "checksum": ""
      },
      {
        "name": "ietf-isis",
        "revision": "2018-12-11",
        "location": [ "https://tiny.cc/ietf-yang/\
                     " ],
        "checksum": ""
      },
      {
        "name": "ietf-interfaces-common",
        "revision": "2018-07-02",
        "location": [ "https://tiny.cc/ietf-yang/\
                     " ],
        "checksum": ""
      },
      {
        "name": "ietf-if-l3-vlan",
        "revision": "2017-10-30",
        "location": [ "https://tiny.cc/ietf-yang/\
                     " ],
        "checksum": ""
      },
      {
```

```
            "name": "ietf-routing-policy",
            "revision": "2018-10-19",
            "location": [ "https://tiny.cc/ietf-yang/\
                        " ],
            "checksum": ""
          },
          {
            "name": "ietf-bgp",
            "revision": "2018-05-09",
            "location": [ "https://tiny.cc/ietf-yang/\
                        " ],
            "checksum": ""
          },
          {
            "name": "ietf-access-control-list",
            "revision": "2018-11-06",
            "location": [ "https://tiny.cc/ietf-yang/\
                        " ],
            "checksum": ""
          }
        ],
        "import-only-module": [
          {
            "name": "ietf-routing-types",
            "revision": "2017-12-04",
            "location": [ "https://tiny.cc/ietf-yang/\
                        ietf-routing-types@2017-12-04.yang" ],
            "checksum": ""
          },
          {
            "name": "iana-routing-types",
            "revision": "2017-12-04",
            "location": [ "https://tiny.cc/ietf-yang/\
                        iana-routing-types@2017-12-04.yang" ],
            "checksum": ""
          },
          {
            "name": "ietf-bgp-types",
            "revision": "2018-05-09",
            "location": [ "https://tiny.cc/ietf-yang/\
                        " ],
            "checksum": ""
          },
          {
            "name": "ietf-packet-fields",
            "revision": "2018-11-06",
            "location": [ "https://tiny.cc/ietf-yang/\
                        " ],
```

```
              "checksum": ""
            },
            {
              "name": "ietf-ethertypes",
              "revision": "2018-11-06",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            }
          ]
        }
      }
    }
  }
  <CODE ENDS>
```

B.3. Package import conflict resolution example

This section provides an example of how a package can resolve conflicting module versions from imported packages.

In this example, YANG package 'example-3-pkg' imports both 'example-import-1' and 'example-import-2' packages. However, the two imported packages implement different versions of 'example-module-A' so the 'example-3-pkg' package selects version '1.2.3' to resolve the conflict. Similarly, for import-only modules, the 'example-3-pkg' package does not require both versions of example-types-module-C to be imported, so it indicates that it only imports revision '2018-11-26' and not '2018-01-01'.

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-import-1-pkg",
    "description": "First imported example package",
    "content-data": {
      "ietf-yang-package:yang-package": {
        "name": "example-import-1",
        "version": "1.0.0",
        "reference": "XXX, draft-rwilton-netmod-yang-packages",
        "revision-date": "2018-01-01",
        "module": [
          {
            "name": "example-module-A",
            "version": "1.0.0"
          },
          {
```

```
              "name": "example-module-B",
              "version": "1.0.0"
            }
          ],
          "import-only-module": [
            {
              "name": "example-types-module-C",
              "revision": "2018-01-01"
            },
            {
              "name": "example-types-module-D",
              "revision": "2018-01-01"
            }
          ]
        }
      }
    }
  }

  {
    "ietf-yang-instance-data:instance-data-set": {
      "name": "example-import-2-pkg",
      "description": "Second imported example package",
      "content-data": {
        "ietf-yang-package:yang-package": {
          "name": "example-import-2",
          "version": "2.0.0",
          "reference": "XXX, draft-rwilton-netmod-yang-packages",
          "revision-date": "2018-11-26",
          "module": [
            {
              "name": "example-module-A",
              "version": "1.2.3"
            },
            {
              "name": "example-module-E",
              "version": "1.1.0"
            }
          ],
          "import-only-module": [
            {
              "name": "example-types-module-C",
              "revision": "2018-11-26"
            },
            {
              "name": "example-types-module-D",
              "revision": "2018-11-26"
            }
```

```
          ]
        }
      }
    }
  }

  {
    "ietf-yang-instance-data:instance-data-set": {
      "name": "example-3-pkg",
      "description": "Importing example package",
      "content-data": {
        "ietf-yang-package:yang-package": {
          "name": "example-3",
          "version": "1.0.0",
          "reference": "XXX, draft-rwilton-netmod-yang-packages",
          "revision-date": "2018-11-26",
          "included-package": [
            {
              "name": "example-import-1",
              "version": "1.0.0"
            },
            {
              "name": "example-import-2",
              "version": "2.0.0"
            }
          ],
          "module": [
            {
              "name": "example-module-A",
              "version": "1.2.3"
            }
          ],
          "import-only-module": [
            {
              "name": "example-types-module-C",
              "revision": "2018-11-26",
              "replaces-revision": [ "2018-01-01 "]
            }
          ]
        }
      }
    }
  }
```

Appendix C. Possible alternative solutions

This section briefly describes some alternative solutions. It can be removed if this document is adopted as a WG draft.

C.1. Using module tags

Module tags have been suggested as an alternative solution, and indeed that can address some of the same requirements as YANG packages but not all of them.

Module tags can be used to group or organize YANG modules. However, this raises the question of where this tag information is stored. Module tags either require that the YANG module files themselves are updated with the module tag information (creating another versioning problem), or for the module tag information to be hosted elsewhere, perhaps in a centralize YANG Catalog, or in instance data documents similar to how YANG packages have been defined in this draft.

One of the principle aims of YANG packages is to be a versioned object that defines a precise set of YANG modules versions that work together. Module tags cannot meet this aim without an explosion of module tags definitions (i.e. a separate module tag must be defined for each package version).

Module tags cannot support the hierachical scheme to construct YANG schema that is proposed in this draft.

C.2. Using YANG library

Another question is whether it is necessary to define new YANG modules to define YANG packages, and whether YANG library could just be reused in an instance data document. The use of YANG packages offers several benefits over just using YANG library:

1. Packages allow schema to be built in a hierarchical fashion. [I-D.ietf-netconf-rfc7895bis] only allows one layer of hierarchy (using module sets), and there must be no conflicts between module revisions in different module-sets.

2. Packages can be made available off the box, with a well defined unique name, avoiding the need for clients to download, and construct/check the entire YANG schema for each device. Instead they can rely on the named packages with secure checksums. YANG library's use of a 'content-id' is unique only to the device that generated them.

3. Packages may be versioned using a semantic versioning scheme, YANG library does not provide a schema level semantic version number.

4. For a YANG library instance data document to contain the necessary information, it probably needs both YANG library and various augmentations (e.g. to include each module's semantic version number), unless a new version of YANG library is defined containing this information. The module definition for a YANG package is specified to contain all of the ncessary information to solve the problem without augmentations

5. YANG library is designed to publish information about the modules, datastores, and datastore schema used by a server. The information required to construct an off box schema is not precisely the same, and hence the definitions might deviate from each other over time.

Author's Address

Robert Wilton
Cisco Systems, Inc.

Email: rwilton@cisco.com

NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 5, 2020

R. Tao
Q. Wu
Huawei

November 2, 2019

YANG Data Node Self Explanation Tags
draft-tao-netmod-yang-node-tags-00

Abstract

This document defines a method to tag data node associated with telemetry data in YANG Modules. This YANG data node tagging method can be used to filter queries of operational state on a server during a "pub/sub" service for YANG datastore updates when the state of all subscriptions of a particular Subscriber to be fetched is huge, so that the amount of data to be streamed out to the destination can be greatly reduced.

An extension statement to be used to indicate YANG data node tags that SHOULD be added by the module implementation automatically(i.e., outside of configuration).

A YANG module [RFC7950] is defined, which augment Module tag model and provides a list of data node entries to allow for adding or removing of data node tags as well as viewing the set of tags associated with a YANG module.

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 https://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 5, 2020.

Copyright Notice

Copyright (c) 2019 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 (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
1.1. Use cases for Data Node tags 3
1.2. Terminology 4
2. Data Node Tag Values 4
2.1. IETF Tags Prefix 4
2.2. Vendor Tags Prefix 4
2.3. User Tags Prefix 5
2.4. Reserved Tags Prefix 5
3. Data Node Tag Management 5
3.1. Module Design Tagging 5
3.2. Implementation Tagging 5
3.3. User Tagging 5
4. Tags Module Structure 6
4.1. Tags Module Tree 6
5. YANG Module 6
6. Guidelines to Model Writers 9
6.1. Define Standard Tags 9
7. IANA Considerations 9
7.1. YANG Data Node Tag Prefixes Registry 9
7.2. IETF YANG Data Node Tags Registry 10
7.3. Updates to the IETF XML Registry 11
7.4. Updates to the YANG Module Names Registry 11
8. Security Considerations 12
9. References 12
9.1. Normative References 12
9.2. Informative References 13
Authors' Addresses 13

1. Introduction

As described [I.D-ietf-netmod-module-tags], the use of tags for classification and organization is fairly ubiquitous not only within IETF protocols, but in the internet itself(e.g., "#hashtags"). A module tag defined in [I.D-ietf-netmod-module-tags] is a string associated only with a module name at module level.

This document define data node tag and associate them with data node within YANG modules. The data node tags can be learnt dynamically by the client from the live server and used to filter queries of configuration or operational state on a server based on these data node tags,.e.g.,return specific object type operational state related to system-management. NETCONF clients can discover data models with data nod tags supported by a NETCONF server via <get-schema> operation. The data node tag capability can also be advertised via Capability Notification Model [I-D.netconf-notification-capabilities] by the NETCONF server or some place where offline document are kept. These tags may be registered as well as assigned during the module definition; assigned by implementations; or dynamically defined and set by users.

This document defines a YANG module [RFC7950] which augments Module tag model and provides a list of data node entries to allow for adding or removing of tags as well as viewing the set of tags associated with a data node within YANG modules.

This document defines an extension statement to be used to indicate tags that SHOULD be added by the module implementation automatically (i.e., outside of configuration).

The YANG data model in this document conforms to the Network Management Datastore Architecture defined in [RFC8342].

1.1. Use cases for Data Node tags

The following is a list of already implemented and potential use cases.

One example use of data node tags would be to help filter different discrete categories of YANG data node within YANG modules supported by a device. For example, if data nodes within YANG modules are suitably tagged and learnt by the client from a live server, then an XPath query can be used by the client to list all of the performance related data nodes supported by a device.

Data node tags can also be used to help coordination when clients are interacting with large amount of devices with the same categories of

YANG data node across different YANG modules. For example, one management client could mark some specific data node across modules implemented in various different devices with the same performance-metric tag, so all the devices can provide consistent representation and reporting for the same category of YANG data nodes.

Future management protocol extensions could allow for filtering queries of configuration or operational state on a server based on tags. For example, return all operational state related to system-management.

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Data Node Tag Values

All data node tags SHOULD begin with a prefix indicating who owns their definition. An IANA registry (Section 7.1) is used to support registering data node tag prefixes. Currently 3 prefixes are defined.

No further structure is imposed by this document on the value following the registered prefix, and the value can contain any YANG type 'string' characters except carriage-returns, newlines and tabs. Therefore, designers, implementers, and users are free to add or not add any structure they may require to their own tag values.

2.1. IETF Tags Prefix

An IETF tag is a data node tag that has the prefix "ietf:dn:". All IETF data node tags are registered with IANA in a registry defined later in this document (Section 7.2).

2.2. Vendor Tags Prefix

A vendor tag is a tag that has the prefix "vendor:dn:". These tags are defined by the vendor that implements the module, and are not registered; however, it is RECOMMENDED that the vendor include extra identification in the tag to avoid collisions such as using the enterpise or organization name following the "vendor:dn:" prefix (e.g., vendor:dn:vendor-defined-classifier).

2.3. User Tags Prefix

A user tag is any tag that has the prefix "user:dn:". These tags are defined by the user/administrator and are not meant to be registered. Users are not required to use the "user:dn:" prefix; however, doing so is RECOMMENDED as it helps avoid prefix collisions.

2.4. Reserved Tags Prefix

Any tag not starting with the prefix "ietf:dn:", "vendor:dn:" or "user:dn:" is reserved for future use. These tag values are not invalid, but simply reserved in the context of specifications (e.g., RFCs).

3. Data Node Tag Management

Tags can become associated with a data node within YANG module in a number of ways. Tags may be defined and associated at module design time, at implementation time without the need of live server, or via user administrative control . As the main consumer of data node tags are users, users may also remove any tag from a live server, no matter how the tag became associated with a data node within a YANG module.

3.1. Module Design Tagging

A data node definition MAY indicate a set of data node tags to be added by the module implementer. These design time tags are indicated using the node-tag extension statement.

If the data node is defined in an IETF standards track document, the data node tags MUST be IETF Tags (2.1). Thus, new data node can drive the addition of new IETF tags to the IANA registry defined in Section 7.2, and the IANA registry can serve as a check against duplication.

3.2. Implementation Tagging

An implementation MAY include additional tags associated with data node within a YANG module. These tags SHOULD be IETF Tags (i.e., registered) or vendor specific tags.

3.3. User Tagging

Data node tags of any kind, with or without a prefix, can be assigned and removed by the user from a live server using normal configuration mechanisms. In order to remove a data node tag from the operational

datastore the user adds a matching "masked-tag" entry for a given data node within the ietf-data-node-tags Module.

4. Tags Module Structure

4.1. Tags Module Tree

The tree associated with the "ietf-data-node-tags" module follows. The meaning of the symbols can be found in [RFC8340].

```
       module: ietf-data-node-tags
       augment /tags:module-tags/tags:module:
         +--rw data-node-tags
            +--rw data-node* [node-name]
               +--rw node-name      nacm:node-instance-identifier
               +--rw tag*           tags:tag
               +--rw masked-tag*    tags:tag
               +--rw group-id       string
```

5. YANG Module

```
<CODE BEGINS> file "ietf-data-node-tags@2019-05-03.yang"
   module ietf-data-node-tags {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-data-node-tags";
     prefix ntags;

     import ietf-yang-types   { prefix yang; }
     import ietf-netconf-acm  { prefix nacm; }
     import ietf-module-tags  { prefix tags; }
     organization
       "IETF NetMod Working Group (NetMod)";
     contact
       "WG Web:  <https://tools.ietf.org/wg/netmod/>
        WG List: <mailto:netmod@ietf.org>

        Author:  Ran Tao
                 <mailto:taoran20@huawei.com>
        Author:  Qin Wu
                 <mailto:bill.wu@huawei.com>";

     // RFC Ed.: replace XXXX with actual RFC number and
     // remove this note.

     description
       "This module describes a mechanism associating tags with YANG data
        node within YANG modules. Tags may be IANA assigned or privately defined
.
        Copyright (c) 2018 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
        (https://trustee.ietf.org/license-info).

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
        NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
        'MAY', and 'OPTIONAL' in this document are to be interpreted as
        described in BCP 14 [RFC2119] [RFC8174] when, and only when,
        they appear in all capitals, as shown here.


        This version of this YANG module is part of RFC XXXX
        (https://tools.ietf.org/html/rfcXXXX); see the RFC itself for
        full legal notices.";

     // RFC Ed.: update the date below with the date of RFC publication
     // and RFC number and remove this note.

     revision 2019-05-03 {
       description
         "Initial revision.";
       reference "RFC XXXX: YANG Data Node Tags";
     }
   typedef node-tag {
       type string {
         length "1..max";
         pattern '[\S ]+';
       }
       description
         "A tag is a type 'string' value that does not include carriage
          return, newline or tab characters. It SHOULD begin with a
          registered prefix; however, tags without a registered prefix
          SHOULD NOT be treated as invalid.";
     }
 extension node-tag {
       argument node-tag;
       description
         "The argument 'tag' is of type 'tag'. This extension statement
          is used by module authors to indicate the data node tags that SHOULD b
e
          added automatically by the system. As such the origin of the
          value for the pre-defined tags should be set to 'system'
          [RFC8342].";
     }
```

```
  augment "/tags:module-tags/tags:module" {
    description
      "Augment the Tags module with data node tag attributes";
     container data-node-tags {
       description
         "Contains the list of data nodes and their associated tags";
       list data-node {
         key "node-name";
         description
           "A list of modules and their associated tags";
         leaf node-name {
           type nacm:node-instance-identifier;
           mandatory true;
           description
             "The YANG module name.";
         }
         leaf-list node-tag {
           type node-tag;
           description
             "Tags associated with the data node within YANG module. See
              the IANA 'YANG Data Node Tag Prefixes' registry for reserved
              prefixes and the IANA'IETF YANG Data Node Tags' registry for
              IETF tags.

              The 'operational' state [RFC8342] view of this list is
              constructed using the following steps:

              1) System tags (i.e., tags of 'system' origin) are added.
              2) User configured tags (i.e., tags of 'intended' origin)
              are added.
              3) Any tag that is equal to a masked-tag is removed.";
         }
         leaf-list node-masked-tag {
           type node-tag;
           description
             "The list of tags that should not be associated with this
              data node. The user can remove (mask) tags from the
              operational state datastore [RFC8342] by adding them to
              this list. It is not an error to add tags to this list
              that are not associated with the data node within the module,
              but they have no operational effect.";
         }
        leaf group-id {
          type string;
          description
           "This group ID is used to identify a set of data nodes
           of the same group which have a common characteristic.";
       }
```

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

6. Guidelines to Model Writers

This section updates [RFC8407].

6.1. Define Standard Tags

A module MAY indicate, using node-tag extension statements, a set of tags that are to be automatically associated with it (i.e., not added through configuration).

```
      module example-module {
        //...
        import module-tags { prefix tags; }
        container top {
        ntags:node-tag "ietf:dn:object-type";
      list X {
        ntags:node-tag "ietf:dn:property";
    }
    container Y {
    ntags:node-tag "ietf:dn:performance-metric";
    }
    }
        // ...
      }
```

The module writer can use existing standard tags, or use new tags defined in the model definition, as appropriate. For IETF standardized modules new data node tags MUST be assigned in the IANA registry defined below, see Section 7.2.

7. IANA Considerations

7.1. YANG Data Node Tag Prefixes Registry

IANA is asked to create a new registry "YANG Data Node Tag Prefixes" grouped under a new "Protocol" category named "YANG Data Node Tag Prefixes".

This registry allocates tag prefixes. All YANG data node tags SHOULD begin with one of the prefixes in this registry.

Prefix entries in this registry should be short strings consisting of lowercase ASCII alpha-numeric characters and a final ":" character.

The allocation policy for this registry is Specification Required [RFC8126]. The Reference and Assignee values should be sufficient to identify and contact the organization that has been allocated the prefix.

The initial values for this registry are as follows.

Prefix	Description	Reference	Assignee
ietf:dn:	IETF Tags allocated in the IANA IETF YANG Data Node Tags registry	[This document]	IETF
vendor:dn:	Non-registered tags allocated by the module implementer.	[This document]	IETF
user:dn:	Non-registered tags allocated by and for the user.	[This document]	IETF

Other standards organizations (SDOs) wishing to allocate their own set of tags should allocate a prefix from this registry.

7.2. IETF YANG Data Node Tags Registry

IANA is asked to create a new registry "IETF YANG Module Tags" grouped under a new "Protocol" category "IETF YANG Module Tags". This registry should be included below "YANG Module Tag Prefixes" when listed on the same page.

This registry allocates tags that have the registered prefix "ietf:". New values should be well considered and not achievable through a combination of already existing IETF tags.

The allocation policy for this registry is IETF Review [RFC8126].

The initial values for this registry are as follows.

Tag	Description	Reference
ietf:dn:object-type	Relates to object type (e.g., interfaces).	[This document]
ietf:dn:performance-metric	Relates to performance metric (e.g., ifstatistics).	[This document]
ietf:dn:property	Represents a object property (e.g.,ifindex).	[This document]
ietf:dn:statistics-operation	Relates to statistics operation(e.g.,average, min, max, sum,etc)	[This document]

7.3. Updates to the IETF XML Registry

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in [RFC3688], the following registration has been made:

URI:
urn:ietf:params:xml:ns:yang:ietf-data-node-tags

Registrant Contact:
The IESG.

XML:
N/A; the requested URI is an XML namespace.

7.4. Updates to the YANG Module Names Registry

This document registers one YANG module in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registration has been made:

```
   name:
      ietf-data-node-tags

   namespace:
      urn:ietf:params:xml:ns:yang:ietf-data-node-tags

   prefix:
      tags

   reference:
      RFC XXXX (RFC Ed.: replace XXX with actual RFC number and remove
      this note.)
```

8. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242].

This document adds the ability to associate data node tag meta-data with YANG modules. This document does not define any actions based on these associations, and none are yet defined, and therefore it does not by itself introduce any new security considerations.

Users of the data node tag-meta data may define various actions to be taken based on the data node tag meta-data. These actions and their definitions are outside the scope of this document. Users will need to consider the security implications of any actions they choose to define.

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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

9.2. Informative References

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc6242>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

Authors' Addresses

Ran Tao
Huawei

Email: taoran20@huawei.com

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

Email: bill.wu@huawei.com

Network Working Group
Internet-Draft
Updates: 7950 (if approved)
Intended status: Standards Track
Expires: April 17, 2020

B. Claise
J. Clarke
R. Rahman
R. Wilton, Ed.
Cisco Systems, Inc.
B. Lengyel
Ericsson
J. Sterne
Nokia
K. D'Souza
AT&T

October 15, 2019

Updated YANG Module Revision Handling
draft-verdt-netmod-yang-module-versioning-01

Abstract

This document specifies a new YANG module update procedure that can document when non-backwards-compatible changes have occurred during the evolution of a YANG module. It extends the YANG import statement with an earliest revision filter to better represent inter-module dependencies. It provides help and guidelines for managing the lifecycle of YANG modules and individual schema nodes. This document updates RFC 7950, RFC 8407 and RFC 8525.

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 https://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 17, 2020.

Copyright Notice

Copyright (c) 2019 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 (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
  1.1. Updates to YANG RFCs 4
2. Terminology and Conventions 4
3. Refinements to YANG revision handling 5
  3.1. Updating a YANG module with a new revision 5
    3.1.1. Backwards-compatible changes 5
    3.1.2. Non-backwards-compatible changes 6
  3.2. nbc-changes revision extension statement 6
  3.3. Revision label 6
  3.4. YANG status description extension statement 7
  3.5. Examples for updating the YANG module revision history . 7
4. Import by derived revision 10
  4.1. Module import examples 11
5. Updates to ietf-yang-library 13
  5.1. Resolving ambiguous module imports 13
  5.2. YANG library versioning augmentations 14
    5.2.1. Advertising revision-label 14
    5.2.2. Reporting how deprecated and obsolete nodes are handled 14
6. Versioning of YANG instance data 15
7. Guidelines for using the YANG module update rules 15
  7.1. Guidelines for YANG module authors 15
    7.1.1. Making non-backwards-compatible changes to a YANG module 16
  7.2. Versioning Considerations for Clients 17
8. Module Versioning Extension YANG Modules 18
9. Contributors 25
10. Security Considerations 26
11. IANA Considerations 26
  11.1. YANG Module Registrations 26
12. References 26

12.1. Normative References 26
12.2. Informative References 27
Appendix A. Appendix 28
A.1. Examples of guidelines for making NBC changes to a YANG module 28
A.1.1. Removing a data node 28
A.1.2. Changing the type of a leaf node 29
A.1.3. Reducing the range of a leaf node 30
A.1.4. Changing the key of a list 30
A.1.5. Renaming a node 31
A.1.6. Changing a default value 32
Authors' Addresses 32

1. Introduction

This document defines a solution to the YANG module lifecycle problems described in [I-D.verdt-netmod-yang-versioning-reqs]. Complementary documents provide a complete solution to the YANG versioning requirements, with the overall relationship of the solution drafts described in [I-D.verdt-netmod-yang-solutions].

Specifically, this document recognises a need (within standards organizations, vendors, and the industry) to sometimes allow YANG modules to evolve with non-backwards-compatible changes, which could cause breakage to clients and importing YANG modules. Accepting that non-backwards-compatible changes do sometimes occur, it is important to have mechanisms to report where these changes occur, and to manage their effect on clients and the broader YANG ecosystem.

The solution comprises five parts:

Refinements to the YANG 1.1 module revision update procedure, supported by new extension statements to indicate when a revision contains non-backwards-compatible changes, and an optional revision label.

A YANG extension statement allowing YANG module imports to specify an earliest module revision that may satisfy the import dependency.

Updates and augmentations to ietf-yang-library to include the revision label in the module descriptions, to report how "deprecated" and "obsolete" nodes are handled by a server, and to clarify how module imports are resolved when multiple versions could otherwise be chosen.

Considerations of how versioning applies to YANG instance data.

Guidelines for how the YANG module update rules defined in this document should be used, along with examples.

Open issues are tracked at <https://github.com/netmod-wg/yang-ver-dt/issues>.

1.1. Updates to YANG RFCs

This document updates [RFC7950] section 11. Section 3 describes modifications to YANG revision handling and update rules, and Section 4 describes a YANG extension statement to do import by derived revision.

This document updates [RFC8525] section 3. Section 5 defines how a client of a YANG library datastore schema chooses which revision of an import-only module is used to resolve a module import when the definition is otherwise ambiguous.

This document updates [RFC8407] section 4.7. Section 7 provides guidelines on managing the lifecycle of YANG modules that may contain non-backwards-compatible changes and a branched revision history.

2. Terminology and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

In addition, this document uses the terminology:

- YANG module revision: An instance of a YANG module, uniquely identified with a revision date, with no implied ordering or backwards compatibility between different revisions of the same module.

- Backwards-compatible (BC) change: A backwards-compatible change between two YANG module revisions, as defined in Section 3.1.1

- Non-backwards-compatible (NBC) change: A non-backwards-compatible change between two YANG module revisions, as defined in Section 3.1.2

3. Refinements to YANG revision handling

[RFC7950] assumes, but does not explicitly state, that the revision history for a YANG module is strictly linear, i.e., it is prohibited to have two independent revisions of a YANG module that are both directly derived from the same parent revision.

This document clarifies [RFC7950] to explicitly allow non linear development of YANG module revisions, so modules MAY have multiple revisions that directly derive from the same parent revision. As per [RFC7950], YANG module revisions continue to be uniquely identified by the module's revision date, and hence all revisions of a module MUST have unique revision dates.

A module's name and revision date identifies a specific immutable definition of that module within its revision history. Hence, if a module includes submodules then the module's "include" statements MUST use "revision-date" substatements to specify the exact revision date of each included submodule.

[RFC7950] section 11 requires that all updates to a YANG module are BC to the previous revision of the module. This document allows for more flexible evolution of YANG modules: NBC changes between module revisions are allowed and are documented using a new "nbc-changes" YANG extension statement in the module revision history.

3.1. Updating a YANG module with a new revision

This section updates [RFC7950] section 11 to refine the rules for permissible changes when a new YANG module revision is created.

Where pragmatic, updates to YANG modules SHOULD be backwards-compatible, following the definition in Section 3.1.1.

A new module revision MAY contain NBC changes, i.e., the semantics of an existing definition MAY be changed in an NBC way without requiring a new definition with a new identifier. A new module revision with NBC changes MUST include the "rev:nbc-changes" extension substatement to signal the potential for incompatibility to existing module users and readers.

3.1.1. Backwards-compatible changes

A change between two module revisions is defined as being "backwards-compatible" if the change conforms to the module update rules specified in [RFC7950] section 11, updated by the following rules:

- A "status" "deprecated" statement MAY be added, or changed from "current" to "deprecated", but adding or changing "status" to "obsolete" is not a backwards-compatible change.

- Obsolete definitions MAY be removed from published modules, and are classified as backwards-compatible changes. In some circumstances it may be helpful to retain the obsolete definitions to ensure that their identifiers are not reused with a different meaning.

- In statements that have any data definition statements as substatements, those data definition substatements MAY be reordered, as long as they do not change the ordering or any "rpc" "input" substatements. If new data definition statements are added, they can be added anywhere in the sequence of existing substatements.

3.1.2. Non-backwards-compatible changes

Any changes to YANG modules that are not defined by Section 3.1.1 as being backwards-compatible are classified as "non-backwards-compatible" changes.

3.2. nbc-changes revision extension statement

The "rev:nbc-changes" extension statement is used to indicate YANG module revisions that contain NBC changes.

If a revision of a YANG module contains changes, relative to the preceding revision in the revision history, that do not conform to the module update rules defined in Section 3.1.1, then a "rev:nbc-changes" extension statement MUST be added as a substatement to the "revision" statement.

Conversely, if a revision does not contain an "rev:nbc-changes" extension substatement then all changes, relative to the preceding revision in the revision history, MUST be backwards-compatible.

3.3. Revision label

Each revision entry in a module or submodule MAY have a revision label associated with it, providing an alternative alias to identify a particular revision of a module or submodule. The revision label could be used to provide an additional versioning identifier associated with the revision.

YANG Semver [I-D.verdt-netmod-yang-semver] defines a versioning scheme based on Semver 2.0.0 [semver] that can be used as a revision

label. All revision labels that match the pattern for the "version" typedef in the ietf-yang-semver YANG module MUST be interpreted as YANG semantic version numbers.

The revision date and revision label within a submodule's revision history have no effect on the including module's revision. Submodules MUST NOT use revision label schemes that could be confused with the including module's revision label scheme.

If a revision has an associated revision label, then it may be used instead of the revision date in two places:

In an "rev:revision-or-derived" extension statement argument.

In the filename of a YANG module, where it takes the form: module-or-submodule-name ['@' revision-label] ('.yang' / '.yin')

3.4. YANG status description extension statement

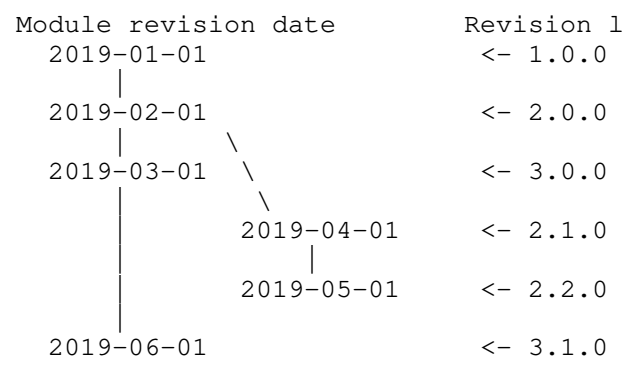
The ietf-yang-revision module specifies the YANG extension statement "status-description" that can be used as a substatement of the status statement. The argument to this extension statement can contain freeform text to help readers of the module understand why the node was deprecated or made obsolete, when it is anticipated that the node will no longer be available for use, and potentially reference other schema elements that can be used instead. An example is shown below.

```
leaf imperial-temperature {
  type int64;
  units "degrees Fahrenheit";
  status deprecated {
    rev:status-description
      "Imperial measurements are being phased out in favor
       of their metric equivalents.  Use metric-temperature
       instead.";
  }
  description
    "Temperature in degrees Fahrenheit.";
}
```

3.5. Examples for updating the YANG module revision history

The following diagram, explanation, and module history illustrates how the branched revision history, "nbc-changes" extension statement, and "revision-label" extension statement could be used:

Example YANG module with branched revision history.



The tree diagram above illustrates how an example module's version history might evolve, over time. For example, the tree might represent the following changes, listed in chronological order from oldest revision to newest:

Example module, revision 2019-06-01:

```
    module example-module {

      namespace "name-space";
      prefix "prefix-name";

      import ietf-yang-revisions { prefix "rev"; }

      description
        "to be completed";

      revision 2019-06-01 {
        rev:revision-label 3.1.0;
        description "Add new functionality.";
      }

      revision 2019-04-01 {
        rev:revision-label 3.0.0;
        rev:nbc-changes;
        description
          "Add new functionality. Remove some deprecated nodes.";
      }

      revision 2019-02-01 {
        rev:revision-label 2.0.0;
        rev:nbc-changes;
        description "Apply bugfix to pattern statement";
      }

      revision 2019-01-01 {
        rev:revision-label 1.0.0;
        description "Initial revision";
      }

      //YANG module definition starts here
```

Example module, revision 2019-05-01:

```
    module example-module {

      namespace "name-space";
      prefix "prefix-name";

      import ietf-yang-revisions { prefix "rev"; }

      description
        "to be completed";

      revision 2019-05-01 {
        rev:revision-label 2.2.0;
        description "Backwards-compatible bugfix to enhancement.";
      }

      revision 2019-03-01 {
        rev:revision-label 2.1.0;
        description "Apply enhancement to older release train.";
      }

      revision 2019-02-01 {
        rev:revision-label 2.0.0;
        rev:nbc-changes;
        description "Apply bugfix to pattern statement";
      }

      revision 2019-01-01 {
        rev:revision-label 1.0.0;
        description "Initial revision";
      }

      //YANG module definition starts here
```

4. Import by derived revision

RFC 7950 allows YANG module "import" statements to optionally require the imported module to have a particular revision date. In practice, importing a module with an exact revision date is often too restrictive because it requires the importing module to be updated whenever any change to the imported module occurs. The alternative choice of using an import statement without any revision date statement is also not ideal because the importing module may not work with all possible revisions of the imported module.

Instead, it is desirable for a importing module to specify a "minimum required revision" of a module that it is compatible with, based on

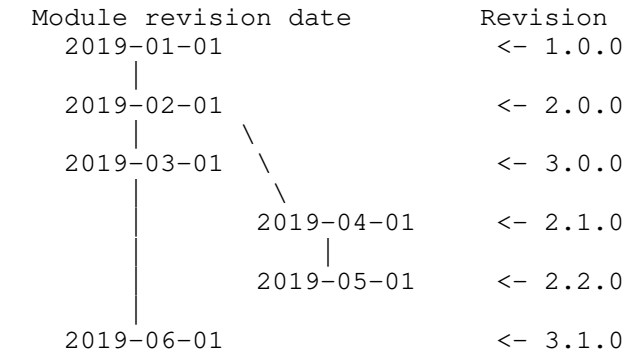
the assumption that later revisions derived from that "minimum required revision" are also likely to be compatible. Many possible changes to a YANG module do not break importing modules, even if the changes themselves are not strictly backwards-compatible. E.g., fixing an incorrect pattern statement or description for a leaf would not break an import, changing the name of a leaf could break an import but frequently would not, but removing a container would break imports if that container is augmented by another module.

The ietf-revisions module defines the "revision-or-derived" extension statement, a substatement to the YANG "import" statement, to allow for a "minium required revision" to be specified during import:

> The argument to the "revision-or-derived" extension statement is a revision date or a revision label.
>
> A particular revision of an imported module satisfies an import's "revision-or-derived" extension statement if the imported module's revision history contains a revision statement with a matching revision date or revision label.
>
> An "import" statement MUST NOT contain both a "revision-or-derived" extension statement and a "revision-date" statement.
>
> The "revision-or-derived" extension statement MAY be specified multiple times, allowing the import to use any module revision that satifies at least one of the "revision-or-derived" extension statements.
>
> The "revision-or-derived" extension statement does not gaurantee that all module revisions that satisfy an import statement are necessarily compatible, it only gives an indication that the revisions are more likely to be compatible. Hence, NBC changes to an imported module may also require new revisions of any importing modules, updated to accommodation those changes, along with updated import "revision-or-derived" extension statements to depend on the updated imported module revision.

4.1. Module import examples

Consider the example module "example-module" from Section 3.5 that is hypothetically available in the following revision/label pairings: 2019-01-01/1.0.0, 2019-02-01/2.0.0, 2019-03-01/3.0.0, 2019-04-01/2.1.0, 2019-05-01/2.2.0 and 2019-06-01/3.1.0. The relationship between the revisions is as before:



4.1.1. Example 1

This example selects module revisions that match, or are derived from the revision 2019-02-01. E.g., this dependency might be used if there was a new container added in revision 2019-02-01 that is augmented by the importing module.It includes revisions/labels: 2019-02-01/2.0.0, 2019-03-01/3.0.0, 2019-04-01/2.1.0, 2019-05-01/2.2.0 and 2019-06-01/3.1.0.

```
import example-module {
  rev:revision-or-derived 2019-02-01;
}
```

Alternatively, the first example could have used the revision label "1.0.0" instead, which selects the same set of revisions/versions.

```
import example-module {
  rev:revision-or-derived 1.0.0;
}
```

4.1.2. Example 2

This example selects module revisions that are derived from 2019-04-01 by using the revision label 2.1.0. It includes revisions/labels: 2019-04-01/2.1.0 and 2019-05-01/2.2.0. Even though 2019-06-01/3.1.0 has a higher revision label version number than 2019-04-01/2.1.0 it is not a derived revision, and hence it is not a valid revision for import.

```
import example-module {
  rev:revision-or-derived 2.1.0;
}
```

4.1.3. Example 3

This example selects revisions derived from either 2019-04-01 or 2019-06-01. It includes revisions/labels: 2019-04-01/2.1.0, 2019-05-01/2.2.0, and 2019-06-01/3.1.0.

```
import example-module {
  rev:revision-or-derived 2019-04-01;
  rev:revision-or-derived 2019-06-01;
}
```

5. Updates to ietf-yang-library

This document updates YANG library [RFC7895] [RFC8525] to clarify how ambiguous module imports are resolved. It also defines the YANG module, ietf-yl-revisions that augments YANG library with new versioning related meta-data.

5.1. Resolving ambiguous module imports

A YANG datastore schema, defined in [RFC8525], can specify multiple revisions of a YANG module in the schema using the "import-only" list, with the requirement from [RFC7950] that only a single revision of a YANG module may be implemented.

If a YANG module import statement does not specify a specific revision within the datastore schema then it could be ambiguous as to which module revision the import statement should resolve to. Hence, a datastore schema constructed by a client using the information contained in YANG library may not exactly match the datastore schema actually used by the server.

The following two rules remove the ambiguity:

If a module import statement could resolve to more than one module revision defined in the datastore schema, and one of those revisions is implemented (i.e., not an "import-only" module), then the import statement MUST resolve to the revision of the module that is defined as being implemented by the datastore schema.

If a module import statement could resolve to more than one module revision defined in the datastore schema, and none of those revisions are implemented, then the import MUST resolve to the module revision with the latest revision date.

5.2. YANG library versioning augmentations

The "ietf-yl-revisions" YANG module has the following structure (using the notation defined in [RFC8340]):

```
module: ietf-yl-revisions
  augment /yanglib:yang-library/yanglib:module-set/yanglib:module:
    +--ro revision-label?   rev:revision-label
  augment /yanglib:yang-library/yanglib:schema:
    +--ro deprecated-nodes-implemented?   empty
    +--ro obsolete-nodes-absent?          empty
```

5.2.1. Advertising revision-label

The ietf-yl-revisions YANG module augments the "module" list in ietf-yang-library with a "revision-label" leaf to optionally declare the revision label associated wth the particular revision of each module.

5.2.2. Reporting how deprecated and obsolete nodes are handled

The ietf-yl-revisions YANG module augments YANG library with two leaves to allow a server to report how it handles status "deprecated" and status "obsolete" nodes. The leaves are:

deprecated-nodes-implemented: If present, this leaf indicates that all schema nodes with a status "deprecated" child statement are implemented equivalently as if they had status "current", or otherwise deviations MUST be used to explicitly remove "deprecated" nodes from the schema. If this leaf is absent then the behavior is unspecified.

obsolete-nodes-absent: If present, this leaf indicates that the server does not implement any status "obsolete" nodes. If this leaf is absent then the behaviour is unspecified.

Servers SHOULD set both the "deprecated-nodes-implemented" and "obsolete-nodes-absent" leaves.

If a server does not set the "deprecated-nodes-implemented" leaf, then clients MUST NOT rely solely on the "rev:nbc-changes" statements to determine whether two module revisions are backwards-compatible, and MUST also consider whether the status of any nodes has changed to "deprecated" and whether those nodes are implemented by the server.

6. Versioning of YANG instance data

Instance data sets [I-D.ietf-netmod-yang-instance-file-format] do not directly make use of the updated revision handling rules described in this document, as compatibility for instance data is undefined.

However, instance data specifies the content-schema of the data-set. This schema SHOULD make use of versioning using revision dates and/or revision labels for the individual YANG modules that comprise the schema or potentially for the entire schema itself (e.g., [I-D.rwilton-netmod-yang-packages]).

In this way, the versioning of a content-schema associated with an instance data set may help a client to determine whether the instance data could also be used in conjunction with other revisions of the YANG schema, or other revisions of the modules that define the schema.

7. Guidelines for using the YANG module update rules

The following text updates section 4.7 of [RFC8407] to revise the guidelines for updating YANG modules.

7.1. Guidelines for YANG module authors

All IETF YANG modules MUST include revision-label statements for all newly published YANG modules, and all newly published revisions of existing YANG modules. The revision-label MUST take the form of a YANG semantic version number [I-D.verdt-netmod-yang-semver].

NBC changes to YANG modules may cause problems to clients, who are consumers of YANG models, and hence YANG module authors are RECOMMENDED to minimize NBC changes and keep changes BC whenever possible.

When NBC changes are introduced, consideration should be given to the impact on clients and YANG module authors SHOULD try to mitigate that impact.

A "rev:nbc-changes" statement SHOULD be added only if there are NBC changes relative to the previous revision.

Removing old revision statements from a module's revision history could break import by revision, and hence it is RECOMMENDED to retain them. If all depencencies have been updated to not import specific revisions of a module, then the corresponding revision statements can be removed from that module. An alternative solution, if the

revision section is too long, would be remove, or curtail, the older description statements associated with the previous revisions.

The "rev:revision-or-derived" extension should be used in YANG module imports to indicate revision dependencies between modules in preference to the "revision-date" statement, which causes overly strict import dependencies and SHOULD NOT be used.

A module that includes submodules MUST use the "revision-date" statement to include specific submodule revisions. Changing a module's include statements to include different submodule revisions requires a new revision of the module.

7.1.1. Making non-backwards-compatible changes to a YANG module

There are various valid situations where a YANG module has to be modified in an NBC way. Here are the different ways in which this can be done:

- NBC changes can be sometimes be done incrementally using the "deprecated" status to provide clients time to adapt to NBC changes.
- NBC changes are done at once, i.e. without using "status" statements. Depending on the change, this may have a big impact on clients.
- If the server can support multiple versions of the YANG module or of YANG packages(as specified in [I-D.rwilton-netmod-yang-packages]), and allows the client to select the version (as per [I-D.wilton-netmod-yang-ver-selection]), then NBC changes MAY be done without using "status" statements. Clients would be required to select the version which they support and the NBC change would have no impact on them

Here are some guidelines on how non-backwards-compatible changes can be made incrementally, with the assumption that deprecated nodes are implemented by the server, and obsolete nodes are not:

1. The changes should be made gradually, e.g. a data node's status SHOULD NOT be changed directly from "current" to "obsolete" (see Section 4.7 of [RFC8407]), instead the status SHOULD first be marked "deprecated" and then when support is removed its status MUST be changed to "obsolete". Instead of using the "obsolete" status, the data node MAY be removed from the model but this has the risk of breaking modules which import the modified module.

2. The new "status-description" extension statement SHOULD be used for nodes which are "obsolete" or "deprecated".

3. For status "deprecated", the "status-description" SHOULD also indicate until when support for the node is guaranteed (if known). If there is a replacement data node, rpc, action or notification for the deprecated node, this SHOULD be stated in the "status-description". The reason for deprecating the node can also be included in the "status-description" if it is deemed to be of potential interest to the user.

4. For status "obsolete", it is RECOMMENDED to keep the "status-description" information, from when the node had status "deprecated, which is still relevant.

5. When obsoleting or deprecating data nodes, the "deprecated" or "obsolete" status SHOULD be applied at the highest possible level in the data tree with an appropriate "status-description" statement. For clarity, the "status" statement SHOULD also be applied to all descendent data nodes, but the "status-description" statement does not need to be repeated if it does not introduce any additional information.

See Appendix A.1 for examples on how NBC changes can be made.

7.2. Versioning Considerations for Clients

Guidelines for clients of modules using the new module revision update procedure:

- Clients SHOULD be liberal when processing data received from a server. For example, the server may have increased the range of an operational node causing the client to receive a value which is outside the range of the YANG model revision it was coded against.

- Clients SHOULD monitor changes to published YANG modules through their revision history, and use appropriate tooling to understand the specific changes between module revision. In particular, clients SHOULD NOT migrate to NBC revisions of a module without understanding any potential impact of the specific NBC changes.

- Clients SHOULD plan to make changes to match published status changes. When a node’s status changes from "current" to "deprecated", clients SHOULD plan to stop using that node in a timely fashion. When a node’s status changes to "obsolete", clients MUST stop using that node.

8. Module Versioning Extension YANG Modules

YANG module with extension statements for annotating NBC changes, revision label, status description, and importing by version.

```
<CODE BEGINS> file "ietf-yang-revisions@2019-09-18.yang"
module ietf-yang-revisions {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-revisions";
  prefix rev;

  import ietf-yang-types {
    prefix yang;
    reference
      "XXXX [ietf-netmod-rfc6991-bis]: Common YANG Data Types";
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     Author:   Benoit Claise
               <mailto:bclaise@cisco.com>

     Author:   Joe Clarke
               <mailto:jclarke@cisco.com>

     Author:   Reshad Rahman
               <mailto:rrahman@cisco.com>

     Author:   Robert Wilton
               <mailto:rwilton@cisco.com>

     Author:   Kevin D'Souza
               <mailto:kd6913@att.com>

     Author:   Balazs Lengyel
               <mailto:balazs.lengyel@ericsson.com>

     Author:   Jason Sterne
               <mailto:jason.sterne@nokia.com>";
  description
    "This YANG 1.1 module contains definitions and extensions to
     support updated YANG revision handling.

     Copyright (c) 2019 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; see
        the RFC itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
        NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
        'MAY', and 'OPTIONAL' in this document are to be interpreted as
        described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
        they appear in all capitals, as shown here.";

    // RFC Ed.: update the date below with the date of RFC publication
    // and remove this note.
    // RFC Ed.: replace XXXX (inc above) with actual RFC number and
    // remove this note.

    revision 2019-09-18 {
      description
        "Initial version.";
      reference
        "XXXX: Updated YANG Module Revision Handling";
    }

    typedef revision-label {
      type string {
        length "1..255";
        pattern '[^\s@]+';
        pattern '\d{4}-\d{2}-\d{2}' {
          modifier invert-match;
        }
      }
      description
        "A label associated with a YANG revision.

         Excludes spaces and '@'.  MUST NOT match revision-date.

         Revision labels that classify as YANG semantic versions,
         as defined by the ietf-yang-semver:version typedef,
         MUST conform to the versioning behaviour defined in
         XXXX [verdt]: YANG Semantic Versioning.";
      reference
```

```
        "XXXX: Updated YANG Module Revision Handling;
         Section 3.3, Revision label";
    }

    typedef revision-date-or-label {
      type union {
        type yang:revision-identifier;
        type revision-label;
      }
      description
        "Represents either a YANG revision date or a revision label";
    }

    extension nbc-changes {
      description
        "This statement is used to indicate YANG module revisions that
         contain non-backwards-compatible changes.

         Each 'revision' statement MAY have a single 'nbc-changes'
         substatement.

         If a revision of a YANG module contains changes, relative to
         the preceding revision in the revision history, that do not
         conform to the module update rules defined in RFC-XXX, then
         the 'nbc-changes' statement MUST be added as a substatement to
         the revision statement.

         Conversely, if a revision of a YANG module only contains
         changes, relative to the preceding revision in the revision
         history, that are classified as 'backwards-compatible' then
         the revision statement MUST NOT contain any 'nbc-changes'
         substatement.";

      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 3.2, nbc-changes revision extension statement";
    }

    extension revision-label {
      argument revision-label;
      description
        "The revision label can be used to provide an additional
         versioning identifier associated with the revision.  E.g., one
         option for a versioning scheme that could be used is [TODO -
         Reference semver draft].

         The format of the revision-label argument MUST conform to the
         pattern defined for the revision-label typedef.
```

```
          Each 'revision' statement MAY have a single 'revision-label'
          substatement.

          Revision labels MUST be unique amongst all revisions of a
          module.";

        reference
          "XXXX: Updated YANG Module Revision Handling;
           Section 3.3, Revision label";
      }

      extension revision-or-derived {
        argument revision-date-or-label;
        description
          "Restricts the revision of the module that may be imported to
           one that matches or is derived from the specified
           revision-date or revision-nlabel.

           The argument value MUST conform to the
           'revision-date-or-label' defined type.

           Each 'import' statement MAY have one or more
           'revision-or-derived' substatements.  If specified multiple
           times, then any module revision that satifies at least one of
           the 'revision-or-derived' statements is an acceptable revision
           for import.

           An 'import' statement MUST NOT contain both a
           'revision-or-derived' extension statement and a
           'revision-date' statement.

           A particular revision of an imported module satisfies an
           import's 'revision-or-derived' extension statement if the
           imported module's revision history contains a revision
           statement with a matching revision date or revision label.

           The 'revision-or-derived' extension statement does not
           gaurantee that all module revisions that satisfy an import
           statement are necessarily compatible, it only gives an
           indication that the revisions are more likely to be
           compatible.";

        reference
          "XXXX: Updated YANG Module Revision Handling;
           Section 4, Import by derived revision";
      }

      extension status-description {
```

```
      argument description;

      description
        "Freeform text that describes why a given node has been
         deprecated or made obsolete.  E.g., the description could be
         used to give the reason for removal, or it could point to an
         alternative schema elements that can be used in lieu of the
         given node.

         Each 'status' statement MAY have a single 'status-description'
         substatement.";

      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 3.4, YANG status description extension statement";
    }
  }
  <CODE ENDS>

  YANG module with augmentations to YANG Library to revision labels

  <CODE BEGINS> file "ietf-yl-revisions@2019-09-18.yang"
  module ietf-yl-revisions {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-yl-revisions";
    prefix yl-rev;

    import ietf-yang-revisions {
      prefix rev;
      reference
        "XXXX: Updated YANG Module Revision Handling";
    }

    import ietf-yang-library {
      prefix yanglib;
      reference "RFC 8525: YANG Libary";
    }

    organization
      "IETF NETMOD (Network Modeling) Working Group";
    contact
      "WG Web:   <https://datatracker.ietf.org/wg/netmod/>
       WG List:  <mailto:netmod@ietf.org>

       Author:   Benoit Claise
                 <mailto:bclaise@cisco.com>

       Author:   Joe Clarke
```

```
                <mailto:jclarke@cisco.com>

        Author:   Reshad Rahman
                  <mailto:rrahman@cisco.com>

        Author:   Robert Wilton
                  <mailto:rwilton@cisco.com>

        Author:   Kevin D'Souza
                  <mailto:kd6913@att.com>

        Author:   Balazs Lengyel
                  <mailto:balazs.lengyel@ericsson.com>

        Author:   Jason Sterne
                  <mailto:jason.sterne@nokia.com>";
    description
      "This module contains augmentations to YANG Library to add module
       level revision label and to provide an indication of how
       deprecated and obsolete nodes are handled by the server.

       Copyright (c) 2019 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; see
       the RFC itself for full legal notices.

       The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
       NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
       'MAY', and 'OPTIONAL' in this document are to be interpreted as
       described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
       they appear in all capitals, as shown here.";

    // RFC Ed.: update the date below with the date of RFC publication
    // and remove this note.
    // RFC Ed.: replace XXXX (including in the imports above) with
    // actual RFC number and remove this note.
    // RFC Ed.: please replace revision-label version with 1.0.0 and
    // remove this note.
    revision 2019-09-18 {
      rev:revision-label 0.1.0;
```

```
      description
        "Initial revision";
      reference
        "XXXX: Updated YANG Module Revision Handling";
    }

    augment "/yanglib:yang-library/yanglib:module-set/yanglib:module" {
      description
        "Augmentation modules with a revision label";
      leaf revision-label {
        type rev:revision-label;
        description
          "The revision label associated with this module revision.
           The label MUST match the rev:label value in the specific
           revision of the module loaded in this module-set.";

        reference
          "XXXX: Updated YANG Module Revision Handling;
           Section 5.2.1, Advertising revision-label";
      }
    }

    augment "/yanglib:yang-library/yanglib:schema" {
      description
        "Augmentations to the ietf-yang-library module to indicate how
         deprecated and obsoleted nodes are handled for each datastore
         schema supported by the server.";

      leaf deprecated-nodes-implemented {
        type empty;
        description
          "If present, this leaf indicates that all schema nodes with a
           status 'deprecated' child statement are implemented
           equivalently as if they had status 'current', or otherwise
           deviations MUST be used to explicitly remove 'deprecated'
           nodes from the schema.  If this leaf is absent then the
           behavior is unspecified.";

        reference
          "XXXX: Updated YANG Module Revision Handling;
           Section 5.2.2, Reporting how deprecated and obsolete nodes
           are handled";
      }

      leaf obsolete-nodes-absent {
        type empty;
        description
          "If present, this leaf indicates that the server does not
```

```
            implement any status 'obsolete' nodes.  If this leaf is
            absent then the behaviour is unspecified.";

          reference
            "XXXX: Updated YANG Module Revision Handling;
             Section 5.2.2, Reporting how deprecated and obsolete nodes
             are handled";
        }
      }
    }
    CODE ENDS>
```

9. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The following individuals are (or have been) members of the design team and have worked on the YANG versioning project:

- Balazs Lengyel
- Benoit Claise
- Ebben Aries
- Jason Sterne
- Joe Clarke
- Juergen Schoenwaelder
- Mahesh Jethanandani
- Michael (Wangzitao)
- Qin Wu
- Reshad Rahman
- Rob Wilton

The initial revision of this document was refactored and built upon [I-D.clacla-netmod-yang-model-update].

Discussons on the use of Semver for YANG versioning has been held with authors of the OpenConfig YANG models. We would like thank both Anees Shaikh and Rob Shakir for their input into this problem space.

10. Security Considerations

The document does not define any new protocol or data model. There are no security impacts.

11. IANA Considerations

11.1. YANG Module Registrations

The following YANG module is requested to be registred in the "IANA Module Names" registry:

The ietf-yang-revisions module:

Name: ietf-yang-revisions

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-revisions

Prefix: rev

Reference: [RFCXXXX]

The ietf-yl-revisions module:

Name: ietf-yl-revisions

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yl-revisions

Prefix: yl-rev

Reference: [RFCXXXX]

12. References

12.1. Normative References

[I-D.ietf-netmod-rfc6991-bis]
Schoenwaelder, J., "Common YANG Data Types", draft-ietf-netmod-rfc6991-bis-01 (work in progress), July 2019.

[I-D.verdt-netmod-yang-semver]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "YANG Semantic Versioning", draft-verdt-netmod-yang-semver-01 (work in progress), October 2019.

[I-D.verdt-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", draft-verdt-netmod-yang-versioning-reqs-02 (work in progress), November 2018.

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

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <https://www.rfc-editor.org/info/rfc7895>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

12.2. Informative References

[I-D.clacla-netmod-yang-model-update]
Claise, B., Clarke, J., Lengyel, B., and K. D'Souza, "New YANG Module Update Procedure", draft-clacla-netmod-yang-model-update-06 (work in progress), July 2018.

[I-D.ietf-netmod-yang-instance-file-format]
Lengyel, B. and B. Claise, "YANG Instance Data File Format", draft-ietf-netmod-yang-instance-file-format-04 (work in progress), August 2019.

[I-D.rwilton-netmod-yang-packages]
Wilton, R., "YANG Packages", draft-rwilton-netmod-yang-packages-01 (work in progress), March 2019.

[I-D.verdt-netmod-yang-solutions]
Wilton, R., "YANG Versioning Solution Overview", draft-verdt-netmod-yang-solutions-01 (work in progress), July 2019.

[I-D.wilton-netmod-yang-ver-selection]
Wilton, R. and R. Rahman, "YANG Schema Version Selection", draft-wilton-netmod-yang-ver-selection-00 (work in progress), March 2019.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[semver] "Semantic Versioning 2.0.0", <https://www.semver.org>.

Appendix A. Appendix

A.1. Examples of guidelines for making NBC changes to a YANG module

Examples of NBC changes include:

- Deleting a data node, or changing it to status obsolete.
- Changing the name, type, or units of a data node.
- Modifying the description in a way that changes the semantic meaning of the data node.
- Any changes that change or reduce the allowed value set of the data node, either through changes in the type definition, or the addition or changes to "must" statements, or changes in the description.
- Adding or modifying "when" statements that reduce when the data node is available in the schema.
- Making the statement conditional on if-feature.

The following sections give guidance for how some of these NBC changes could be made to a YANG module:

A.1.1. Removing a data node

Removing a leaf or container from the data tree, e.g. because support for the corresponding feature is being removed:

1. The node's status is changed to "deprecated" and it is supported for at least one year. This is a BC change.

2. When the node is not available anymore, its status is changed to "obsolete" and the "status-description" updated, this is an NBC change. The "status-description" is used to explain why the node is not available anymore.

If the server can support NBC versions of the YANG module simultaneously using version selection, then the changes can be done immediately:

1. The new revision of the YANG module has the node's status changed to "obsolete" and the "status-description" updated, this is an NBC change.

2. Clients which require the data node select the older module revision

A.1.2. Changing the type of a leaf node

Changing the type of a leaf-node. e.g. consider a "vpn-id" node of type integer being changed to a string:

1. The status of node "vpn-id" is changed to "deprecated" and the node should be available for at least one year. This is a BC change.

2. A new node, e.g. "vpn-name", of type string is added to the same location as the existing node "vpn-id". This new node has status "current" and its description explains that it is replacing node "vpn-id".

3. During the period of time where both nodes are available, how the server behaves when either node is set is outside the scope of this document and will vary on a case by case basis. Here are some options:

    1. A server may prevent the new node from being set if the old node is already set (and vice-versa). The new node may have a when statement to achieve this. The old node must not have a when statement since this would be an NBC change, but the server could reject the old node from being set if the new node is already set.

    2. If the new node is set and a client does a get or get-config operation on the old node, the server could map the value. For example, if the new node "vpn-name" has value "123" then

the server could return integer value 123 for the old node "vpn-id". However, if the value can not be mapped, we need a way of returning "unsupported" TBD.

4. When node "vpn-id" is not available anymore, its status is changed to "obsolete" and the "status-description" is updated. This is an NBC change.

If the server can support NBC versions of the YANG module simultaneously using version selection, then the changes can be done immediately:

1. In the new revision of the YANG module, the status of node "vpn-id" is changed to "obsolete". This is an NBC change.

2. New node "vpn-name" is added to the same location as described above.

3. Clients which require the data node select the older module revision

4. A server should not map between the nodes "vpn-id" and "vpn-name", i.e. if a client creates a data instance with "vpn-name" then that data instance should not be visible to a client using a module revision which has "vpn-id" (and vice-versa).

A.1.3. Reducing the range of a leaf node

Reducing the range of values of a leaf-node. e.g. consider a "vpn-id" node of type integer being changed from type uint32 to type uint16:

1. If all values which are being removed were never supported, e.g. if a vpn-id of 65536 or higher was never accepted, this is a BC change for the functionality (no functionality change). Even if it is an NBC change for the YANG model, there should be no impact for clients using that YANG model.

2. If one or more values being removed was previously supported, e.g. if a vpn-id of 65536 was accepted previously, this is an NBC change for the YANG model. Clients using the old YANG model will be impacted, so a change of this nature should be done carefully, e.g. by using the steps described in Appendix A.1.2

A.1.4. Changing the key of a list

Changing the key of a list has a big impact to the client. For example, consider a "sessions" list which has a key "interface" and

there is a need to change the key to "dest-address", such a change can be done in steps:

1. The status of list "sessions" is changed to "deprecated" and the list should be available for at least one year. This is a BC change.

2. A new list is created in the same location with the same data but with "dest-address" as key. Finding an appropriate name for the new list can be tricky especially if the name of the existing list was perfect. In this case the new list is called "sessions-address", has status "current" and its description should explain that it is replacing list "session".

3. During the period of time where both lists are available, how the server behaves when either list is set is outside the scope of this document and will vary on a case by case basis. Here are some options:

    1. A server could prevent the new list from being set if the old list already has entries (and vice-versa).

    2. If the new list is set and a client does a get or get-config operation on the old list, the server could map the entries. However if the new list has entries which would lead to duplicate keys in the old list, the mapping can not be done.

4. When list "sessions" is not available anymore, its status is changed to "obsolete" and the "status-description" is updated. This is an NBC change.

If the server can support NBC versions of the YANG module simultaneously using version selection, then the changes can be done immediately:

1. The new revision of the YANG module has the list "sessions" modified to have "dest-address" as key, this is an NBC change.

2. Clients which require the previous functionality select the older module revision

A.1.5. Renaming a node

A leaf-node or a container may be renamed, either due to a spelling error in the previous name or because of a better name. For example a node "ip-adress" could be renamed to "ip-address":

1. The status of the existing node "ip-adress" is changed to "deprecated" and the node should be available for at least one year. This is a BC change.

2. The new node "ip-address" is added to the same location as the existing node "ip-adress". This new node has status "current" and its description should explain that it is replacing node "ip-adress".

3. During the period of time where both nodes are available, how the server behaves when either node is set is outside the scope of this document and will vary on a case by case basis. Here are some options:

    1. A server could prevent the new node from being set if the old node is already set (and vice-versa). The new node could have a when statement to achieve this. The old node must not have a when statement since this would be an NBC change, but the server could reject the old node from being set if the new node is already set.

    2. If the new node is set and a client does a get or get-config operation on the old node, the server could use the value of the new node. For example, if the new node "ip-address" has value X then the server may return value X for the old node "ip-adress".

4. When node "ip-adress" is not available anymore, its status is changed to "obsolete" and the "status-description" is updated. This is an NBC change.

If the server can support NBC versions of the YANG module simultaneously using version selection, then the changes can be done immediately:

1. The new revision of the YANG module has the node with the new name replacing the node with the old name, this is an NBC change.

2. Clients which require the previous node name select the older module revision

A.1.6. Changing a default value

Authors' Addresses

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Phone: +32 2 704 5622
Email: bclaise@cisco.com

Joe Clarke
Cisco Systems, Inc.
7200-12 Kit Creek Rd
Research Triangle Park, North Carolina
United States of America

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

Reshad Rahman
Cisco Systems, Inc.

Email: rrahman@cisco.com

Robert Wilton (editor)
Cisco Systems, Inc.

Email: rwilton@cisco.com

Balazs Lengyel
Ericsson
Magyar Tudosok Korutja
1117 Budapest
Hungary

Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

Jason Sterne
Nokia

Email: jason.sterne@nokia.com

Kevin D'Souza
AT&T
200 S. Laurel Ave
Middletown, NJ
United States of America

Email: kd6913@att.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 19, 2020

R. Wilton
Cisco Systems, Inc.

November 16, 2019

YANG Schema Comparison
draft-verdt-netmod-yang-schema-comparison-00

Abstract

This document specifies an algorithm for comparing two revisions of a YANG schema to determine the scope of changes, and a list of changes, between the revisions. The output of the algorithm can be used to help select an appropriate revision-label or YANG semantic version number for a new revision. This document defines a YANG extension that provides YANG annotations to help the tool accurately determine the scope of changes between two revisions.

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 https://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, 2020.

Copyright Notice

Copyright (c) 2019 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 (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

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

Table of Contents

1. Introduction 2
2. Terminology and Conventions 4
3. Generic YANG schema tree comparison algorithm 4
3.1. YANG module revision scope extension annotations 5
4. YANG module comparison algorithm 5
5. YANG schema comparison algorithms 6
5.1. Standard YANG schema comparison algorithm 6
5.2. Filtered YANG schema comparison algorithm 6
6. Comparison tooling 7
7. Module Versioning Extension YANG Modules 7
8. Contributors 10
9. Security Considerations 11
10. IANA Considerations 11
10.1. YANG Module Registrations 11
11. References 11
11.1. Normative References 11
11.2. Informative References 12
Author's Address 13

1. Introduction

Warning, this is an early (-00) draft with the intention of scoping the outline of the solution, hopefully for the WG to back the direction of the solution. Refinement of the solution details is expected, if this approach is accepted by the WG.

This document defines a solution to Requirement 2.2 in [I-D.verdt-netmod-yang-versioning-reqs]. Complementary documents provide a complete solution to the YANG versioning requirements, with the overall relationship of the solution drafts described in [I-D.verdt-netmod-yang-solutions].

YANG module 'revision-labels' [I-D.verdt-netmod-yang-module-versioning] and the use of YANG semantic version numbers [I-D.verdt-netmod-yang-semver] can be used to help manage and report changes between revisions of individual YANG modules.

YANG packages [I-D.rwilton-netmod-yang-packages] along with YANG semantic version numbers can be used to help manage and report changes between revisions of YANG schema.

[I-D.verdt-netmod-yang-module-versioning] and
[I-D.rwilton-netmod-yang-packages] define how to classify changes between two module or package revisions, respectively, as backwards compatible or non-backwards-compatible.
[I-D.verdt-netmod-yang-semver] refines the definition, to allow backwards compatible changes to be classified as 'minor changes' or 'editorial changes'.

'Revision-label's and YANG semantic version numbers, whilst being generally simple and helpful in the mainline revision history case, are not sufficient in all scenarios. For example, when comparing two revisions/versions on independent revision branches, without a direct ancestor relationship between the two revisions/versions. In this cases, an algorithmic comparison approach is beneficial.

In addition, the module revision history's 'nbc-changes' extension statement, and YANG semantic version numbers, effectively declare the worst case scenario. If any non-backwards-compatible changes are restricted to only parts of the module/schema that are not used by an operator, then the operator is able to upgrade, and effectively treat the differences between the two revisions/versions as backwards compatible because they are not materially impacted by the non-backwards-compatible changes.

Hence, this document defines algorithms that can be applied to revisions of YANG modules or versions of YANG schema (e.g., as represented by YANG packages), to determine the changes, and scope of changes between the revisions/versions.

For many YANG statements, programmatic tooling can determine whether the changes between the statements constitutes a backwards-compatible or non-backwards-compatible change. However, for some statements, it is not feasible for current tooling to determine whether the changes are backwards-compatible or not. For example, in the general case, tooling cannot determine whether the change in a YANG description statement causes a change in the semantics of a YANG data node. If the change is to fix a typo or spelling mistake then the change can be classified as an editorial backwards-compatible change. Conversely, if the change modifies the behavioral specification of the data node then the change would need to be classified as either a non editorial backwards-compatible change or a non-backwards-compatible change. Hence, extension statements are defined to annotate a YANG module with additional information to clarify the scope of changes in cases that cannot be determined by algorithmic comparison.

Open issues are tracked at <https://github.com/netmod-wg/yang-ver-dt/issues>, tagged with 'schema-comparison'.

2. Terminology and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses terminology introduced in the YANG versioning requirements document [I-D.verdt-netmod-yang-versioning-reqs].

This document makes of the following terminology introduced in the YANG Packages [I-D.rwilton-netmod-yang-packages]:

- YANG schema

In addition, this document defines the terminology:

- Change scope: Whether a change between two revisions is classified as non-backwards-compatible, backwards-compatible, or editorial.

3. Generic YANG schema tree comparison algorithm

The generic schema comparison algorithm works on any YANG schema. This could be a schema associated with an individual YANG module, or a YANG schema represented by a set of modules, e.g., specified by a YANG package.

The algorithm performs a recursive tree wise comparison of two revisions of a YANG schema, with the following behavior:

> The comparison algorithm primarily acts on the parts of the schema defined by unique identifiers.
>
> Each identifier is qualified with the name of the module that defines the identifier.
>
> Identifiers in different namespaces (as defined in 6.2.1 or RFC 7950) are compared separately. E.g., 'features' are compared separately from 'identities'.
>
> Within an identifier namespace, the identifiers are compared between the two schema revisions by qualified identifier name. The 'renamed-from' extension allow for a meaningful comparison where the name of the identifier has changed between revisions. The 'renamed-from' identifier parameter is only used when an identifier in the new schema revision cannot be found in the old schema revision.

YANG extensions, features, identities, typedefs are checked by comparing the properties defined by their YANG sub-statements between the two revisions.

YANG groupings, top-level data definition statements, rpcs, and notifications are checked by comparing the top level properties defined by their direct child YANG sub-statements, and also by recursively checking the data definition statements.

The rules specified in section 3 of [I-D.verdt-netmod-yang-module-versioning] determine whether the changes are backwards-compatible or non-backwards-compatible.

The rules specified in section 3.2 of [I-D.rwilton-netmod-yang-packages] determine whether backwards-compatible changes are 'minor' or 'editorial'.

For YANG description", "must", and "when" statements, the "backwards-compatible" and "editorial" extension statements can be used to mark instances when the statements have changed in a backwards-compatible or editorial way. Since by default the comparison algorithm assumes that any changes in these statements are non-backwards-compatible. XXX, more info required here, since the revisions in the module history probably need to be available for this to work in the general branched revisions case.

Submodules are not relevant for schema comparison purposes, i.e. the comparison is performed after submodule resolution has been completed.

3.1. YANG module revision scope extension annotations

4. YANG module comparison algorithm

The schema comparison algorithm defined in Section 3 can be used to compare the schema for individual modules, but with the following modifications:

Changes to the module's metadata information (i.e. module level description, contact, organization, reference) should be checked (as potential editorial changes).

The module's revision history should be ignored from the comparison.

Changes to augmentations and deviations should be sorted by path and compared.

5. YANG schema comparison algorithms

5.1. Standard YANG schema comparison algorithm

The standard method for comparing two YANG schema versions is to individually compare the module revisions for each module implemented by the schema using the algorithm defined in Section 4 and then aggregating the results together:

- If all implemented modules in the schema have only changed in an editorial way then the schema is changed in an editorial way
- If all implemented modules in the schema have only been changed in an editorial or backwards-compatible way then the schema is changed in a backwards-compatible way
- Otherwise if any implemented module in the schema has been changed in a non-backwards-compatible way then the schema is changed in a non-backwards-compatible way.

The standard schema comparison method is the RECOMMENDED scheme to calculate the version number change for new versions of YANG packages, because it allows the package version to be calculated based on changes to implemented modules revision history (or YANG semantic version number if used to identify module revisions).

5.2. Filtered YANG schema comparison algorithm

Another method to compare YANG schema, that is less likely to report inconsequential differences, is to construct full schema trees for the two schema versions, directly apply a version of the comparison algorithm defined in Section 3. This may be particular useful when the schema represents a complete datastore schema for a server because it allows various filtered to the comparison algorithm to provide a more specific answer about what changes may impact a particular client.

The full schema tree can easily be constructed from a YANG package definition, or alternative YANG schema definition.

Controlled by input parameters to the comparison algorithm, the following parts of the schema trees can optionally be filtered during the comparison:

All "grouping" statements can be ignored (after all "use" statements have been processed when constructing the schema).

All module and submodule metadata information (i.e. module level description, contact, organization, reference) can be ignored.

The comparison can be restricted to the set of features that are of interest (different sets of features may apply to each schema versions).

The comparison can be restricted to the subset of data nodes, RPCs, notifications and actions, that are of interest (e.g., the subset actually used by a particular client), providing a more meaningful result.

The comparison could filter out backwards-compatible 'editorial' changes.

In addition to reporting the overall scope of changes at the schema level, the algorithm output can also optionally generate a list of specific changes between the two schema, along with the classification of those individual changes.

6. Comparison tooling

'pyang' has some support for comparison two module revisions, but this is currently limited to a linear module history.

TODO, it would be helpful if there is reference tooling for schema comparison.

7. Module Versioning Extension YANG Modules

YANG module with extension statements for annotating NBC changes, revision label, status description, and importing by version.

```
<CODE BEGINS> file "ietf-yang-rev-annotations@2019-11-11.yang"
module ietf-yang-rev-annotations {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-rev-annotations";
  prefix rev-ext;

  organization
    "IETF NETMOD (Network Modeling) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     Author:   Robert Wilton
               <mailto:rwilton@cisco.com>";
```

```
  description
    "This YANG 1.1 module contains extensions to annotation to YANG
     module with additional metadata information on the nature of
     changes between two YANG module revisions.

     XXX, maybe these annotations could also be included in
     ietf-yang-revisions?

     Copyright (c) 2019 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; see
     the RFC itself for full legal notices.

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
     NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
     'MAY', and 'OPTIONAL' in this document are to be interpreted as
     described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
     they appear in all capitals, as shown here.";

  // RFC Ed.: update the date below with the date of RFC publication
  // and remove this note.
  // RFC Ed.: replace XXXX (inc above) with actual RFC number and
  // remove this note.

  revision 2019-11-11 {
    description
      "Initial version.";
    reference
      "XXXX: YANG Schema Comparison";
  }

  extension backwards-compatible {
    argument revision-date-or-label;
    description
      "Identifies a revision (by revision-label, or revision date if
       a revision-label is not available) where a
       backwards-compatible change has occurred relative to the
       previous revision listed in the revision history.

       The format of the revision-label argument MUST conform to the
```

```
          pattern defined for the ietf-yang-revisions
          revision-date-or-label typedef.

          The following YANG statements MAY have zero or more
          'rev-ext:non-backwards-compatible' statements:
              description
              must
              when

          Each YANG statement MUST only a have a single
          non-backwards-compatible, backwards-compatible, or editorial
          extension statement for a particular revision-label, or
          corresponding revision-date.";

        reference
          "XXXX: YANG Schema Comparison;
           Section XXX, XXX";
      }

      extension editorial {
        argument revision-date-or-label;
        description
          "Identifies a revision (by revision-label, or revision date if
           a revision-label is not available) where an editorial change
           has occurred relative to the previous revision listed in the
           revision history.

           The format of the revision-label argument MUST conform to the
           pattern defined for the ietf-yang-revisions
           revision-date-or-label typedef.

           The following YANG statements MAY have zero or more
           'rev-ext:non-backwards-compatible' statements:
               description

           Each YANG statement MUST only a have a single
           non-backwards-compatible, backwards-compatible, or editorial
           extension statement for a particular revision-label or
           corresponding revision-date.";

        reference
          "XXXX: YANG Schema Comparison;
           Section XXX, XXX";
      }

      extension renamed-from {
        argument yang-identifier;
        description
```

```
          "Specifies a previous name for this identifier.

           This can be used when comparing schema to optimize handling
           for data nodes that have been renamed rather than naively
           treated them as data nodes that have been deleted and
           recreated.

           The argument 'yang-identifier' MUST take the form of a YANG
           identifier, as defined in section 6.2 of RFC 7950.

           Any YANG statement that takes a YANG identifier as its
           argument MAY have a single 'rev-ext:renamed-from'
           sub-statement.

           TODO, we should also facilitate identifiers being moved into
           other modules, e.g. by supporting a module-name qualified
           identifier.";

        reference
          "XXXX: YANG Schema Comparison;
           Section XXX, XXX";
      }
    }
    <CODE ENDS>
```

8. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The following individuals are (or have been) members of the design team and have worked on the YANG versioning project:

- Balazs Lengyel
- Benoit Claise
- Bo Wu
- Ebben Aries
- Jason Sterne
- Joe Clarke
- Juergen Schoenwaelder
- Mahesh Jethanandani

- Michael Wang
- Qin Wu
- Reshad Rahman
- Rob Wilton

The ideas for a tooling based comparison of YANG module revisions was first described in [I-D.clacla-netmod-yang-model-update]. This document extends upon those initial ideas.

9. Security Considerations

The document does not define any new protocol or data model. There are no security impacts.

10. IANA Considerations

10.1. YANG Module Registrations

The following YANG module is requested to be registered in the "IANA Module Names" registry:

The ietf-yang-rev-annotations module:

Name: ietf-yang-rev-annotations

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-rev-annotations

Prefix: rev-ext

Reference: [RFCXXXX]

11. References

11.1. Normative References

[I-D.rwilton-netmod-yang-packages]
Wilton, R., "YANG Packages", draft-rwilton-netmod-yang-packages-02 (work in progress), October 2019.

[I-D.verdt-netmod-yang-module-versioning]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "Updated YANG Module Revision Handling", draft-verdt-netmod-yang-module-versioning-01 (work in progress), October 2019.

[I-D.verdt-netmod-yang-semver]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "YANG Semantic Versioning", draft-verdt-netmod-yang-semver-01 (work in progress), October 2019.

[I-D.verdt-netmod-yang-solutions]
Wilton, R., "YANG Versioning Solution Overview", draft-verdt-netmod-yang-solutions-01 (work in progress), July 2019.

[I-D.verdt-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", draft-verdt-netmod-yang-versioning-reqs-02 (work in progress), November 2018.

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

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <https://www.rfc-editor.org/info/rfc7895>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

11.2. Informative References

[I-D.clacla-netmod-yang-model-update]
Claise, B., Clarke, J., Lengyel, B., and K. D'Souza, "New YANG Module Update Procedure", draft-clacla-netmod-yang-model-update-06 (work in progress), July 2018.

[I-D.ietf-netmod-yang-instance-file-format]
Lengyel, B. and B. Claise, "YANG Instance Data File Format", draft-ietf-netmod-yang-instance-file-format-04 (work in progress), August 2019.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[semver] "Semantic Versioning 2.0.0", <https://www.semver.org>.

Author's Address

Robert Wilton
Cisco Systems, Inc.

Email: rwilton@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 13, 2020

B. Claise
J. Clarke, Ed.
R. Rahman
R. Wilton, Ed.
Cisco Systems, Inc.
B. Lengyel
Ericsson
J. Sterne
Nokia
K. D'Souza
AT&T

October 11, 2019

YANG Semantic Versioning
draft-verdt-netmod-yang-semver-01

Abstract

This document specifies a scheme for applying a modified set of semantic versioning rules to revisions of YANG modules. Additionally, this document defines a revision label for this modified semver scheme based on the specification in draft-verdt-netmod-yang-module-versioning.

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 https://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 13, 2020.

Copyright Notice

Copyright (c) 2019 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 (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
2. Terminology and Conventions 3
3. YANG Semantic Versioning 3
3.1. YANG Semantic Versioning Pattern 3
3.2. Semantic Versioning Scheme for YANG Artifacts 4
3.2.1. Examples for YANG semantic version numbers 6
3.3. YANG Semantic Version Update Rules 7
3.4. Examples of the YANG Semver Label 8
3.4.1. Example Module Using YANG Semver 8
3.4.2. Example of Package Using YANG Semver 10
4. Import Module by Semantic Version 10
5. YANG Module 11
6. Contributors 12
7. Security Considerations 13
8. IANA Considerations 13
9. References 13
9.1. Normative References 13
9.2. Informative References 14
Authors' Addresses 14

1. Introduction

[I-D.verdt-netmod-yang-module-versioning] puts forth a number of concepts relating to modified rules for updating modules, a means to signal when a new revision of a module has non-backwards-compatible (NBC) changes compared to its previous revision, and a versioning scheme that uses the revision history as a lineage for determining from where a specific revision of a YANG module is derived. Additionally, section 3.3 of [I-D.verdt-netmod-yang-module-versioning] defines a revision label which can be used as an overlay or alias to provide additional context or an additional way to refer to a specific revision.

This document defines a labeling scheme that uses modified [semver] rules for YANG artifacts (i.e., YANG modules and YANG packages [I-D.rwilton-netmod-yang-packages]) as well as the revision label

definition for using this scheme. The goal of this is to add a human readble version label that provides compatibility information for the YANG artifact without one needing to compare or parse its body. The label and rules defined herein represent the RECOMMENDED revision label scheme for IETF YANG artifacts.

2. Terminology and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Additionally, this document uses the following terminology:

- YANG artifact: YANG modules, YANG packages [I-D.rwilton-netmod-yang-packages], and YANG schema elements are examples of YANG artifacts for the purposes of this document.

3. YANG Semantic Versioning

This section defines YANG Semantic Versioning, explains how it is used with YANG artifacts, and the rules associated with changing a artifact's semantic version number when its contents are updated.

3.1. YANG Semantic Versioning Pattern

YANG artifacts that employ semantic versioning MUST use a version string (e.g., in revision-label or as a package version) that corresponds to the following pattern: X.Y.Zv. Where:

- X, Y and Z are mandatory non-negative integers that are each less than 32768 and MUST NOT contain leading zeroes

- The '.' is a literal period (ASCII character 0x2e)

- v is an optional single character modifier that MUST be either 'm' or 'M' if it is specified

Additionally, [semver] defines two specific types of metadata that may be appended to a semantic version string. Pre-release metadata MAY be appended to a semver string after a trailing '-' character. Build metadata MAY be appended after a trailing '+' character. If both pre-release and build metadata are present, then build metadata MUST follow pre-release metadata. These optional elements MUST be ignored by YANG semver parsers, but are allowed in order to support

all of the [semver] rules. Thus, a version lineage that follows strict [semver] rules is allowed for a YANG artifact.

Other version schemes MUST NOT use version strings that match this same pattern. For example, they may choose to use leading characters to distinguish themselves from YANG semver.

A YANG module is defined in this document which contains single typedef that formally specifies this version pattern.

3.2. Semantic Versioning Scheme for YANG Artifacts

This document defines the YANG semantic versioning scheme that is used for YANG artifacts that employ the semver label. The versioning scheme has the following properties:

The YANG semantic versioning scheme is extended from version 2.0.0 of the semantic versioning scheme defined at semver.org [semver] to cover the additional requirements for the management of YANG artifact lifecyles that cannot be addressed using the semver.org 2.0.0 versioning scheme alone.

Unlike the [semver] versioning scheme, the YANG semantic versioning scheme supports limited updates to older versions of YANG artifacts, to allow for bug fixes and enhancements to artifact versions that are not the latest. However, it does not provide for the unlimited branching and updating of older revisions which are documented by the general rules in [I-D.verdt-netmod-yang-module-versioning].

YANG artifacts that follow the [semver] versioning scheme are fully compatible with implementations that understand the YANG semantic versioning scheme defined in this document.

If updates are always restricted to the latest revision of the artifact only, then the version numbers used by the YANG semantic versioning scheme are exactly the same as those defined by the [semver] versioning scheme.

Every YANG module versioned using the YANG semantic versioning scheme specifies the module's semantic version number as the argument to the 'rev:revision-label' statement.

Because the rules put forth in [I-D.verdt-netmod-yang-module-versioning] are designed to work well with existing versions of YANG and allow for artifact authors to migrate to this scheme, it is not expected that all revisions of a given YANG artifact will have a semantic version label. For example,

the first revision of a module may have been produced before this scheme was available.

YANG packages that make use of this semantic versioning scheme will have their semantic version as the value of the "revision_label" property.

As stated above, the YANG semver version number is expressed as a string of the form: 'X.Y.Zv'; where X, Y, and Z each represent non-negative integers smaller than 32768 without leading zeroes, and v represents an optional single character suffix: 'm' or 'M'.

- 'X' is the MAJOR version. Changes in the major version number indicate changes that are non-backwards-compatible to versions with a lower major version number.

- 'Y' is the MINOR version. Changes in the minor version number indicate changes that are backwards-compatible to versions with the same major version number, but a lower minor version number and no patch 'm' or 'M' modifier.

- 'Zv' is the PATCH version and modifier. Changes in the patch version number can indicate editorial, backwards-compatible, or non-backwards-compatible changes relative to versions with the same major and minor version numbers, but lower patch version number, depending on what form modifier 'v' takes:

  * If the modifier letter is absent, the change represents an editorial change. An editorial change is defined to be a change in the YANG artifact's content that does not affect the semantic meaning or functionality provided by the artifact in any way. An example is correcting a spelling mistake in the description of a leaf within a YANG module. Note: restructuring how a module uses, or does not use, submodules is treated as an editorial level change on the condition that there is no change in the module's semantic behavior due to the restructuring.

  * If, however, the modifier letter is present, the meaning is described below:

  * 'm' - the change represents a backwards-compatible change

  * 'M' - the change represents a non-backwards-compatible change

The YANG articact name and YANG semantic version number uniquely identifies a revision of said artifact. There MUST NOT be multiple instances of a YANG artifact definition with the same name and YANG

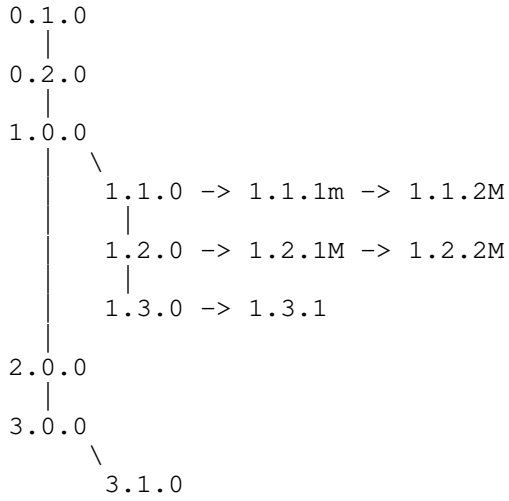
semantic version number but different content (and in the case of modules, different revision dates).

There MUST NOT be multiple versions of a YANG artifact that have the same MAJOR, MINOR and PATCH version numbers, but different patch modifier letters. E.g., artifact version "1.2.3M" MUST NOT be defined if artifact version "1.2.3" has already been defined.

3.2.1. Examples for YANG semantic version numbers

The following diagram and explanation illustrates how YANG semantic version numbers work.

Example YANG semantic version numbers for an example artifact:



Assume the tree diagram above illustrates how an example YANG module's version history might evolve. For example, the tree might represent the following changes, listed in chronological order from oldest revision to newest:

0.1.0 - first beta module version

0.2.0 - second beta module version (with NBC changes)

1.0.0 - first release (may have NBC changes from 0.2.0)

1.1.0 - added new functionality, leaf "foo" (BC)

1.2.0 - added new functionality, leaf "baz" (BC)

1.3.0 - improve existing functionality, added leaf "foo-64" (BC)

1.3.1 - improve description wording for "foo-64" (Editorial)

1.1.1m - backport "foo-64" leaf to 1.1.x to avoid implementing "baz" from 1.2.0 (BC)

2.0.0 - change existing model for performance reasons, e.g. re-key list (NBC)

1.1.2M - NBC point bug fix, not required in 2.0.0 due to model changes (NBC)

3.0.0 - NBC bugfix, rename "baz" to "bar"; also add new BC leaf "wibble"; (NBC)

1.2.1M - backport NBC fix, changing "baz" to "bar"

1.2.2M - backport "wibble". This is a BC change but "M" modifier is sticky.

3.1.0 - introduce new leaf "wobble" (BC)

The partial ordering relationships based on the semantic versioning numbers can be defined as follows:

1.0.0 < 1.1.0 < 1.2.0 < 1.3.0 < 2.0.0 < 3.0.0 < 3.1.0

1.0.0 < 1.1.0 < 1.1.1m < 1.1.2M

1.0.0 < 1.1.0 < 1.2.0 < 1.2.1M < 1.2.2M

There is no ordering relationship between 1.1.1M and either 1.2.0 or 1.2.1M, except that they share the common ancestor of 1.1.0.

Looking at the version number alone, the module definition in 2.0.0 does not necessarily contain the contents of 1.3.0. However, the module revision history in 2.0.0 may well indicate that it was edited from module version 1.3.0.

3.3. YANG Semantic Version Update Rules

When a new revision of an artifact is produced, then the following rules define how the YANG semantic version number for the new artifact revision is calculated, based on the changes between the two artifact revisions, and the YANG semantic version number of the base artifact revision from which the changes are derived:

1. If an artifact is being updated in a non-backwards-compatible way, then the artifact version "X.Y.Z[m|M]" MUST be updated to "X+1.0.0" unless that artifact version has already been defined with different content, in which case the artifact version "X.Y.Z+1M" MUST be used instead.

2. If an artifact is being updated in a backwards-compatible way, then the next version number depends on the format of the current version number:

   i "X.Y.Z" - the artifact version MUST be updated to "X.Y+1.0", unless that artifact version has already been defined with different content, when the artifact version MUST be updated to "X.Y.Z+1m" instead.

   ii "X.Y.Zm" - the artifact version MUST be updated to "X.Y.Z+1m".

   iii "X.Y.ZM" - the artifact version MUST be updated to "X.Y.Z+1M".

3. If an artifact is being updated in an editorial way, then the next version number depends on the format of the current version number:

   i "X.Y.Z" - the artifact version MUST be updated to "X.Y.Z+1"

   ii "X.Y.Zm" - the artifact version MUST be updated to "X.Y.Z+1m".

   iii "X.Y.ZM" - the artifact version MUST be updated to "X.Y.Z+1M".

4. YANG artifact semantic version numbers beginning with 0, i.e "0.X.Y" are regarded as beta definitions and need not follow the rules above. Either the MINOR or PATCH version numbers may be updated, regardless of whether the changes are non-backwards-compatible, backwards-compatible, or editorial.

3.4. Examples of the YANG Semver Label

3.4.1. Example Module Using YANG Semver

Below is a sample YANG module that uses the YANG semver revision label based on the rules defined in this document.

```
module yang-module-name {
```

```
        namespace "name-space";
        prefix "prefix-name";

        import ietf-yang-revisions { prefix "rev"; }

        description
          "to be completed";

        revision 2018-02-28 {
          description "Added leaf 'wobble'";
          rev:revision-label "3.1.0";
        }

        revision 2017-12-31 {
          description "Rename 'baz' to 'bar', added leaf 'wibble'";
          rev:revision-label "3.0.0";
          rev:nbc-changes;
        }

        revision 2017-10-30 {
          description "Change the module structure";
          rev:revision-label "2.0.0";
          rev:nbc-changes;
        }

        revision 2017-08-30 {
          description "Clarified description of 'foo-64' leaf";
          rev:revision-label "1.3.1";
        }

        revision 2017-07-30 {
          description "Added leaf foo-64";
          rev:revision-label "1.3.0";
        }

        revision 2017-04-20 {
          description "Add new functionality, leaf 'baz'";
          rev:revision-label "1.2.0";
        }

        revision 2017-04-03 {
          description "Add new functionality, leaf 'foo'";
          rev:revision-label "1.1.0";
        }

        revision 2017-04-03 {
          description "First release version.";
          rev:revision-label "1.0.0";
```

```
        }

        // Note: semver rules do not apply to 0.X.Y labels.

        revision 2017-01-30 {
          description "NBC changes to initial revision";
          semver:module-version "0.2.0";
        }

        revision 2017-01-26 {
          description "Initial module version";
          semver:module-version "0.1.0";
        }

        //YANG module definition starts here
```

3.4.2. Example of Package Using YANG Semver

Below is an example YANG package that uses the semver revision label based on the rules defined in this document.

```
   {
     "ietf-yang-instance-data:instance-data-set": {
       "name": "example-yang-pkg",
       "target-ptr": "TBD",
       "timestamp": "2018-09-06T17:00:00Z",
       "description": "Example IETF package definition",
       "content-data": {
         "ietf-yang-package:yang-package": {
           "name": "example-yang-pkg",
           "version": "1.3.1",
           ...
}
```

4. Import Module by Semantic Version

[I-D.verdt-netmod-yang-module-versioning] allows for imports to be done based on a module or a derived revision of a module. The rev:revision-or-derived statement can specify either a revision date or a revision label. When importing by semver, the YANG semver revision label value MAY be used as an argument to rev:revision-or-derived. In so, any module which has that semver label as its latest revision label or has that label in its revision history can be used to satisfy the import requirement. For example:

```
        import example-module {
          rev:revision-or-derived "3.0.0";
        }
```

Note: the import lookup does not stop when a non-backward-compatible change is encountered. That is, if module B imports a module A at or derived from version 2.0.0, resolving that import will pass through a revision of module A with version 2.1.0M in order to determine if the present instance of module A derives from 2.0.0.

5. YANG Module

This YANG module contains the typedef for the YANG semantic version.

```
<CODE BEGINS> file "ietf-yang-semver@2019-09-06.yang"
  module ietf-yang-semver {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-yang-semver";
    prefix yangver;

    organization
      "IETF NETMOD (Network Modeling) Working Group";
    contact
      "WG Web:   <http://tools.ietf.org/wg/netmod/>
       WG List:  <mailto:netmod@ietf.org>

       Author:   Joe Clarke
                 <mailto:jclarke@cisco.com>";
    description
      "This module provides type and grouping definitions for YANG
       packages.

       Copyright (c) 2019 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; see
       the RFC itself for full legal notices.";

    // RFC Ed.: update the date below with the date of RFC publication
    // and remove this note.
    // RFC Ed.: replace XXXX with actual RFC number and remove this
    // note.

    revision 2019-09-06 {
```

```
      description
        "Initial revision";
      reference
        "RFC XXXX: YANG Semantic Versioning.";
    }

    /*
     * Typedefs
     */

    typedef version {
      type string {
        pattern '\d+[.]\d+[.]\d+[mM]?(-[\w\d.]+)?([+][\w\d\.]+)?';
      }
      description
        "Represents a YANG semantic version number.  Note:
         additional rules apply to the dot-separated numeric identifiers
         which are spelled out in the reference for this typedef.";
      reference
        "RFC XXXX: YANG Semantic Versioning.";
    }
  }
    <CODE ENDS>
```

6. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The design team consists of the following members whom have worked on the YANG versioning project:

- Balazs Lengyel
- Benoit Claise
- Ebben Aries
- Jason Sterne
- Joe Clarke
- Juergen Schoenwaelder
- Mahesh Jethanandani
- Michael (Wangzitao)
- Qin Wu

- Reshad Rahman
- Rob Wilton

The initial revision of this document was refactored and built upon [I-D.clacla-netmod-yang-model-update].

Discussons on the use of Semver for YANG versioning has been held with authors of the OpenConfig YANG models based on their own [openconfigsemver]. We would like thank both Anees Shaikh and Rob Shakir for their input into this problem space.

7. Security Considerations

The document does not define any new protocol or data model. There are no security impacts.

8. IANA Considerations

None.

9. References

9.1. Normative References

[I-D.verdt-netmod-yang-module-versioning]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "Updated YANG Module Revision Handling", draft-verdt-netmod-yang-module-versioning-00 (work in progress), July 2019.

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

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <https://www.rfc-editor.org/info/rfc7895>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

9.2. Informative References

[I-D.clacla-netmod-yang-model-update]
Claise, B., Clarke, J., Lengyel, B., and K. D'Souza, "New YANG Module Update Procedure", draft-clacla-netmod-yang-model-update-06 (work in progress), July 2018.

[I-D.ietf-netmod-yang-instance-file-format]
Lengyel, B. and B. Claise, "YANG Instance Data File Format", draft-ietf-netmod-yang-instance-file-format-04 (work in progress), August 2019.

[I-D.rwilton-netmod-yang-packages]
Wilton, R., "YANG Packages", draft-rwilton-netmod-yang-packages-01 (work in progress), March 2019.

[openconfigsemver]
"Semantic Versioning for Openconfig Models", <http://www.openconfig.net/docs/semver/>.

[semver] "Semantic Versioning 2.0.0", <https://www.semver.org>.

Authors' Addresses

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Phone: +32 2 704 5622
Email: bclaise@cisco.com

Joe Clarke (editor)
Cisco Systems, Inc.
7200-12 Kit Creek Rd
Research Triangle Park, North Carolina
United States of America

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

Reshad Rahman
Cisco Systems, Inc.

Email: rrahman@cisco.com

Robert Wilton (editor)
Cisco Systems, Inc.

Email: rwilton@cisco.com

Balazs Lengyel
Ericsson
Magyar Tudosok Korutja
1117 Budapest
Hungary

Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

Jason Sterne
Nokia

Email: jason.sterne@nokia.com

Kevin D'Souza
AT&T
200 S. Laurel Ave
Middletown, NJ
United States of America

Email: kd6913@att.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 6, 2020

R. Wilton, Ed.
Cisco Systems, Inc.

November 3, 2019

YANG Versioning Solution Overview
draft-verdt-netmod-yang-solutions-02

Abstract

This document gives a brief overview of the different drafts that comprise a full solution to the YANG versioning requirements draft. The purpose of this draft is to help readers understand how the discrete parts of the YANG versioning solution fit together during working group development of the solution drafts.

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 https://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 6, 2020.

Copyright Notice

Copyright (c) 2019 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 (https://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. Solution Drafts 2
2.1. Updated YANG Module Revision Handling 3
2.2. Module semantic version number scheme 4
2.3. Versioned YANG packages 4
2.4. Protocol operations for package version selection 5
2.5. YANG schema comparison tooling 5
3. Contributors 6
4. Security Considerations 6
5. IANA Considerations 7
6. References 7
6.1. Normative References 7
6.2. Informative References 7
Author's Address 7

1. Introduction

[I-D.ietf-netmod-yang-versioning-reqs] documents the requirements for any solution to the YANG [RFC7950] versioning problem. Chapter 5 lists the formal requirements that a complete solution requires.

The aim of this draft is to help readers understand how the different solution drafts fit together, and also which drafts contribute solutions to particular individual requirements. The overall solution comprises five individual drafts:

1. [I-D.verdt-netmod-yang-module-versioning]

2. [I-D.verdt-netmod-yang-semver]

3. [I-D.rwilton-netmod-yang-packages]

4. [I-D.wilton-netmod-yang-ver-selection]

5. YANG schema comparison tooling (not yet published)

Open issues, across all of the solution drafts are tracked at <https://github.com/netmod-wg/yang-ver-dt/issues>.

2. Solution Drafts

The complete solution to the YANG versioning requirements comprises five solution drafts, that are summarized below.

2.1. Updated YANG Module Revision Handling

In summary, [I-D.verdt-netmod-yang-module-versioning] specifies minimal extensions and updates to the YANG language, YANG Library, and YANG author guidelines to provide more flexible YANG module revision handling. The intent is that these changes and extensions could be folded into future revisions of the updated specifications. The draft provides a solution for all requirements except Req 2.2, Req 3.1 and Req 3.2.

The extensions and changes in the draft can be summarized thus:

- It defines a YANG extension statement to indicate where non-backwards-compatible changes have occurred in a module's revision history.

- It relaxes the rules for the module revision history to allow for a non-linear module revision history. I.e., any given module revision may have multiple revisions directly derived from it.

- It defines a new import extension statement that restricts the allowed module revisions that satisfy the import to only those derived from a specified module revision.

- It defines a revision label extension statement to allow an informative name to be associated with a particular revision date, and to be used in import statements, YANG module filenames, and is available in YANG library. One example of how the revision label could be used is to associate a semantic versioning scheme to YANG module revisions.

- It updates the YANG rules for changes between module revisions that are allowed to be classified as backwards-compatible. In particular, marking a node as obsolete is no longer classified as a backwards compatible change.

- It provides updated guidance on how servers handle deprecated and obsolete YANG nodes and augments YANG library with additional leaves to report the server's behavior to clients.

- It provides an extension statement to allow a description statement to be associated with a YANG status statement, providing more information about why the status has changed.

- It defines how versioning relates to YANG instance data.

- It refines the guidelines for updating modules, taking into consideration that non-backwards-compatible changes are sometimes necessary for various pragmatic reasons.

2.2. Module semantic version number scheme

[I-D.verdt-netmod-yang-semver] defines a semantic versioning scheme derived from the semver.org 2.0.0 specification that can be used in conjunction with the revision label extension statement defined in Section 2.1 to allow semantic version numbers to be used to manage the revision lifecycle of YANG modules. This draft provides an enhanced solution for Req 2.1, but organizations authoring modules are not obliged to use this specific versioning scheme, and could choose a different overlaid versioning scheme, or none at all and rely solely on revision dates.

The aims of the YANG semantic versioning scheme are:

To generally allow clients to determine whether NBC changes have occurred between two revisions from the version number alone, without having to check the full revision history.

To give a more informative identifier for a branched revision history over revision dates alone.

To allow revision branches that contain fixes for published non-latest releases.

2.3. Versioned YANG packages

The two previous drafts address version and revision management of individual modules. [I-D.rwilton-netmod-yang-packages] provides a mechanism to group a set of related YANG modules revisions together, into a construct called a YANG package, and to apply a version scheme to the group.

The core part of this draft are YANG module definitions that define a YANG package, that are used as an augmentation to YANG library, and also in YANG instance data documents for offline access.

The principle aims of the packages draft are:

To provide an alternative simpler mechanism to manage conformance of modules. Rather than checking conformance against a set of individual YANG module revisions, it should be easier to check for conformance against a much smaller set of YANG package versions.

To provide an easier mechanism for clients to check conformance with a server. Rather that downloading and comparing all individual module revisions, the client can just check whether the package version is compatible. The package definition could be retrieved and cached from multiple sources.

The YANG packages draft does not address any of the versioning requirements directly, but provides the foundation building blocks for the version selection solution, described in Section 2.4, that addresses Reqs 3.1 and 3.2.

2.4. Protocol operations for package version selection

[I-D.wilton-netmod-yang-ver-selection] specifies a solution for requirements 3.1 and 3.2 via the use of [I-D.rwilton-netmod-yang-packages] and a protocol based version selection scheme that can be used by clients to choose a particular YANG datastore schema from the set of datastore schema that are supported by the server.

The version selection optionally allows:

Servers to support a single, selectable YANG package at a particular version, that is used for all NETCONF/RESTCONF interactions.

Servers to support multiple selectable YANG packages and package versions, with different clients able to concurrently access different packages and different package versions.

2.5. YANG schema comparison tooling

A tooling based solution is proposed for Req 2.2, that allows two YANG schema versions to be algorithmically compared, with the algorithm reporting the list of differences between the two YANG schema and whether each change is regarded as being backwards-compatible, or non-backwards-compatible. Annotations to the YANG modules, via the use of extension statements, may help improve the accuracy of the comparison algorithm, particularly for statements that are very hard to algorithmically classify the scope of any differences (e.g., a change in the semantic behaviour of a data node defined via modifications to the associated YANG description statement). Given that Req 2.2 is a softer requirement, and practical experience with the tooling is required, it is proposed that this work is deferred at this time.

When comparing a module schema, a tool would also be able to take into account enabled features, deviations, and the subset of the

schema being used by the client. This would allow a tooling based approach to give a more accurate answer as to whether a client would be affected when upgrading between two software versions, than looking at the revision history, or comparing semantic version numbers.

3. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The following individuals are (or have been) members of that design team and have contributed to defining the problem, specifying the requirements, and working on a solution:

- Balazs Lengyel
- Benoit Claise
- Ebben Aries
- Jason Sterne
- Joe Clarke
- Juergen Schoenwaelder
- Mahesh Jethanandani
- Michael (Wangzitao)
- Qin Wu
- Reshad Rahman
- Rob Wilton
- Susan Hares
- Wu Bo

4. Security Considerations

The document does not define any new protocol or data model. There is no security impact.

5. IANA Considerations

None

6. References

6.1. Normative References

[I-D.ietf-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", draft-ietf-netmod-yang-versioning-reqs-01 (work in progress), July 2019.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

6.2. Informative References

[I-D.rwilton-netmod-yang-packages]
Wilton, R., "YANG Packages", draft-rwilton-netmod-yang-packages-02 (work in progress), October 2019.

[I-D.verdt-netmod-yang-module-versioning]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "Updated YANG Module Revision Handling", draft-verdt-netmod-yang-module-versioning-01 (work in progress), October 2019.

[I-D.verdt-netmod-yang-semver]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "YANG Semantic Versioning", draft-verdt-netmod-yang-semver-01 (work in progress), October 2019.

[I-D.wilton-netmod-yang-ver-selection]
Wilton, R., Rahman, R., and J. Clarke, "YANG Schema Version Selection", draft-wilton-netmod-yang-ver-selection-01 (work in progress), November 2019.

Author's Address

Robert Wilton (editor)
Cisco Systems, Inc.

Email: rwilton@cisco.com

Network Working Group
Internet-Draft
Updates: 6241,8040 (if approved)
Intended status: Standards Track
Expires: May 4, 2020

R. Wilton
R. Rahman
J. Clarke
Cisco Systems, Inc.

November 1, 2019

YANG Schema Version Selection
draft-wilton-netmod-yang-ver-selection-01

Abstract

This document defines protocol mechanisms to allow clients, using NETCONF or RESTCONF, to choose which YANG schema to use for interactions with a server, out of the available YANG schemas supported by a server. The provided functionality allow servers to support clients in a backwards compatible way, at the same time allowing for non-backwards-compatible updates to YANG modules.

This draft provides a solution to YANG versioning requirements 3.1 and 3.2.

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 https://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 4, 2020.

Copyright Notice

Copyright (c) 2019 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 (https://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. Terminology and Conventions 2
2. Introduction 3
3. Background 4
4. Objectives 4
5. Solution Overview 5
5.1. NETCONF 6
5.1.1. New capability to advertise schema sets supported . . 6
5.1.2. <select-schema-sets> operation 7
5.2. RESTCONF 8
6. Version selection from a server perspective 8
7. Version selection from a client's perspective 9
8. Limitations of the solution 10
9. Schema Version Selection YANG module 10
10. YANG Module 11
11. Security Considerations 16
12. IANA Considerations 16
12.1. NETCONF Capability URNs 16
13. Open Questions/Issues 16
14. Acknowledgements 16
15. References 17
15.1. Normative References 17
15.2. Informative References 18
Authors' Addresses 19

1. Terminology and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses terminology introduced in the YANG versioning requirements document [I-D.verdt-netmod-yang-versioning-reqs], the YANG Module Versioning document [I-D.verdt-netmod-yang-module-versioning] and the YANg Packages document [I-D.rwilton-netmod-yang-packages].

This document also makes of the following terminology introduced in the Network Management Datastore Architecture [RFC8342]:

- datastore schema

In addition, this document makes use of the following terminology:

- YANG schema: The combined set of schema nodes for a set of YANG module revisions, taking into consideration any deviations and enabled features.

- versioned schema: A YANG schema with an associated YANG semantic version number, e.g., as might be described by a YANG package[I-D.rwilton-netmod-yang-packages].

- schema set: A set of related versioned YANG schema, one for each datastore that is supported.

TODO - 'schema' and 'versioned schema' could be defined in the packages draft.

2. Introduction

This document describes how NETCONF and RESTCONF clients can choose a particular YANG schema they wish to choose to interact with a server.

[I-D.verdt-netmod-yang-versioning-reqs] defines requirements that any solution to YANG versioning must have.

[I-D.verdt-netmod-yang-semver], which is based on [I-D.verdt-netmod-yang-module-versioning], specifies a partial solution to the YANG versioning requirements that focuses on using semantic versioning within individual YANG modules, but does not address all the requirements listed in the requirements document. Of particular relevance here, requirements 3.1 and 3.2 are not addressed.

[I-D.rwilton-netmod-yang-packages] describes how sets of related YANG modules can be grouped together into a logical entity that is versioned using the YANG semantic versioning number scheme. Different packages can be defined for different sets of YANG modules, e.g., packages could be defined for the IETF YANG modules, OpenConfig YANG modules, a vendor's YANG modules. Different versions of these package definitions can be defined as the contents of these packages evolve over time, and as the versions of the YANG modules included in the package evolve.

This document defines how YANG packages can be used to represent versioned datastore schema, and how clients can choose which versioned schemas to use during interactions with a device.

3. Background

There are three ways that the lifecycle of a data model can be managed:

1. Disallow all non-backwards-compatible updates to a YANG module. Broadly this is the approach adopted by [RFC7950], but it has been shown to be too inflexible in some cases. E.g. it makes it hard to fix bugs in a clean fashion - it is not clear that allowing two independent data nodes (one deprecated, one current) to configure the same underlying property is robustly backwards compatible in all scenarios, particularly if the value space and/or default values differ between the module revisions.

2. Allow non-backwards-compatible updates to YANG modules, and use a mechanism such as semantic version numbers to communicate the likely impact of any changes to module users, but require that clients handle non-backwards-compatible changes in servers by migrating to new versions of the modules. Without version selection, this is what the [I-D.verdt-netmod-yang-semver] approach likely achieves.

3. Allow non-backwards-compatible updates to YANG modules, but also provide mechanisms to allow servers to support multiple versions of YANG modules, and provide clients with some ability to select which versions of YANG modules they wish to interact with, subject to some reasonable constraints. This is the approach that this document aims to address. It is worth noting that the idea of supporting multiple versions of an API is not new in the wider software industry, and there any many examples of where this approach has been successfully used.

4. Objectives

The goals of the schema version selection document are:

- To provide a mechanism where non-backwards-compatible changes and bug fixes can be made to YANG modules without forcing clients to immediately migrate to new versions of those modules as they get implemented.

- To allow servers to support multiple versions of a particular YANG schema, and to allow clients to choose which YANG schema version to use when interoperating with the server. The aim here is to give operators more flexibility as to when they update their software.

- To provide a mechanism to allow different YANG schema families (e.g., SDO models, OpenConfig models, Vendor models) to be supported by a server, and to allow clients to choose which YANG schema family is used to interoperate with the server.

The following points are non objective of this document:

- This document does not provide a mechanism to allow clients to choose arbitrary sets of YANG module versions to interoperate with the server.

- Servers are not required to concurrently support clients using different YANG schema families or versioned schema. A server MAY choose to only allow a single schema family or single versioned schema to be used by all clients.

- There is no requirement for a server to support every published version of a YANG package, particularly if some package versions are backwards compatible. Clients are required to interoperate with backwards compatible updates of YANG modules. E.g., if a particular package was available in versions 1.0.0, 1.1.0, 1.2.0, 2.0.0, 3.0.0 and 3.1.0, then a server may choose to only support versions 1.2.0, 2.0.0, and 3.1.0, with the knowledge that all clients should be able to interoperate with the server.

- There is no requirement to support all parts of all versioned schemas. For some nbc changes in modules, it is not possible for a server to support both the old and new module versions, and to convert between the two. Where appropriate deviations can be used, and otherwise an out of band mechanism is used to indicate where a mapping has failed.

5. Solution Overview

An overview of the solution is as follows:

1. YANG packages, specified in [I-D.rwilton-netmod-yang-packages], are defined for the different versioned schema supported by a server:

    * Separate packages can be defined for different families of schema, e.g., SDO, OpenConfig, or vendor native.

    * Separate packages can be defined for each versioned schema within a schema family.

    * Separate packages may be defined for different datastores, if the datastores use different datastore schema. For example, a

different datastore schema, and hence package, might be used for <operational> vs the conventional datastores.

2. Each server advertises, via an operational data model:

   * All of the YANG packages that may be used during version selection. The packages can also be made available for offline consumption via instance data documents, as described in [I-D.rwilton-netmod-yang-packages].

   * Grouped sets of versioned schema, where each set defines the versioned schema used by each supported datastore, and each versioned schema is represented by a YANG package instance.

3. Each server supports the operations to:

   * Allow a client to configure which schema version set to use for the default NETCONF/RESTCONF connections.

   * Allow a client to configure additional separate RESTCONF protocol instances, which use different schema version sets on those protocol instances. See Section 5.2.

   * Allow a client using NETCONF to use the "select-schema-sets" RPC to choose which schema sets it wants to use for the lifetime of the current NECONF session. See Section 5.1.

4. The server internally maps requests between the different protocol instances to the internal device implementation.

5.1. NETCONF

5.1.1. New capability to advertise schema sets supported

A new NETCONF :schema-sets capability, using base:1.1 defined in [RFC6241], is used to indicate what schema sets the server is willing to support. The server sends an unordered comma separated list (with no white spaces) of schema sets it supports. A server MAY concurrently support clients using different YANG package versions.

The :schema-sets capability is identified by the following capability string:

```
urn:ietf:params:netconf:capability:schema-sets:1.0
```

In this example, the server advertises its (abbreviated) <hello> as follows. This indicates that the server supports the following schema sets:

example-ietf-routing: Versions 2.1.0 and 1.3.1

example-vendor-xxx: Versions 9.2.3 and 8.4.2

Some extra white spaces have been added for display purposes only.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
<capability>urn:ietf:params:netconf:base:1.1</capability>
<capability>
urn:ietf:params:netconf:capability:schema-sets:1.0?list=
example-ietf-routing@2.1.0,example-ietf-routing@1.3.1,
example-vendor-xxx@9.2.3,example-vendor-xxx@8.4.2
</capability>
</capabilities>
</hello>
```

TODO - add mechanism to indicate default version

5.1.2. <select-schema-sets> operation

Description: Used by a client to select schema-sets which have been advertised by the server via the mechanism described above in Section 5.1.1. The schema-sets are selected for the lifetime of the NETCONF session, unless new schema-sets are subsequently selected via this operation.

Parameters:

schema-set: Schema-set(s)that the client wants to use for this session.

Positive response: if the server was able to satisfy the request, an <rpc-reply> is sent that includes an <ok> element.

Negative response: An <rpc-error> element is included in the <rpc-reply> if the request cannot be completed for any reason. A <select-schema-sets> operation can fail for a number of reasons, such as a YANG package version is not supported by the server, or a different version of a YANG package has already been selected by another client.

Example: a client selects schema sets example-ietf-routing version 1.3.1 and example-vendor-xxx version 9.2.3:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <select-schema-sets>
    <schema-sets>
      <schema-set>example-ietf-routing@1.3.1</schema-set>
      <schema-set>example-vendor-xxx@9.2.3</schema-set>
    </schema-sets>
  </select-schema-sets>
</rpc>

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/> <!-- Schema set selection succeeded -->
</rpc-reply>
```

TODO - add error indication

5.2. RESTCONF

To enable servers which can support different versions of YANG packages:

- On servers, a unique RESTCONF root resource can be configured to map to each schema-set
- A default schema-set can be configured.

Clients select the desired schema-set by choosing the corresponding RESTCONF root resource

This updates Section 3.1 of [RFC8040]. For example, consider a device which has been configured with root-path "/restconf/example-ietf-routing-1.3.1" for secondary schema set "example-ietf-routing-1.3.1". Any operations by the client on schema set "example-ietf-routing-1.3.1" would use "/restconf/example-ietf-routing-1.3.1" as the RESTCONF root resource.

6. Version selection from a server perspective

The general premise of this solution is that servers generally implement one native schema, and the version selection scheme is used to support older version of that native schema and also foreign schema specified by external entities.

Overall the solution relies on the ability to map instance data between different schema versions. Depending on the scope of difference between the schema versions then some of these mappings

may be very hard, or even impossible, to implement. Hence, there is still a strong incentive to try and minimize nbc changes between schema versions to minimize the mapping complexity.

Server implementations MUST serialize configuration requests across the different schema. The expectation is that this would be achieved by mapping all requests to the device's native schema version.

Datastore validation needs to be performed in two places, firstly in whichever schema a client is interacting in, and secondly in the native schema for the device. This could have a negative performance impact.

Depending on the complexity of the mappings between schema versions, it may be necessary for the mappings to be stateful.

TODO - Figure out how hot fixes that slightly modify the schema are handled.

7. Version selection from a client's perspective

Clients can use configuration to choose which schema sets are available.

Clients cannot choose arbitrary individual YANG module versions, and are instead constrained by the versions that the server makes available.

Each client protocol connection is to one particular schema set. From that client session perspective it appears as if the client is interacting with a regular server. If the client queries YANG library that the version of YANG Library that is returned matches the schema set that is being used for that server instance.

The server may not support a schema with the exact version desired by the client, and they have to accept a later version that is backwards compatible with their desired version. Clients may also have to accept later schema versions that contain NBC fixes, although the assumption is that such nbc fixes should be designed to minimize the impact on clients.

There is no guarantee that servers will always be able to support all older schema versions. Deviations should be used where necessary to indicate that the server is unable to faithfully implement the older schema version.

If clients interact with a server using multiple versions, they should not exact that all data nodes in later module versions can

always be backported to older schema versions. TODO - Specify how mapping errors can be reported to client.

8. Limitations of the solution

Not all schema conversions are possible. E.g. an impossible type conversion, or something has been removed. The solution is fundamentally limited by how the schemas actually change, this solution does not provide a magic bullet that can solve all versioning issues.

9. Schema Version Selection YANG module

The YANG schema version selection YANG module is used by a device to report the schema-sets that are available, and to allow clients to choose which schema-set they wish to use.

Feature are used to allow servers to decide whether they allow the primary schema-set to be changed, and/or allow secondary schema-sets to be configured.

The primary schema-set is the datastore schema reported by YANG Library.

If secondary schema-sets are configured:

With NETCONF, the "select-schema-sets" RPC is used by the client to chosse which schema set(s) it wants to use for the current NETCONF session.

With RESTCONF, the configured root path prefix is used by the client for a particular schema set.

Different schema-sets may support different datastores.

The "ietf-schema-version-selection" YANG module has the following structure:

```
module: ietf-schema-version-selection
  +--rw schema-selection
     +--rw schema-sets* [name]
     |  +--rw name          string
     |  +--ro datastores* [datastore]
     |     +--ro datastore    ds:datastore-ref
     |     +--ro packages* [package]
     |        +--ro package    -> /yanglib:yang-library/pkg:package/name
     |        +--ro version?   -> /yanglib:yang-library/pkg:package[pkg:name = c
urrent()
     |                              /../package]/version
     +--rw restconf {secondary-schema-set}?
     |  +--rw schema-sets* [schema-set]
     |     +--rw schema-set    -> /schema-selection/schema-sets/name
     |     +--rw root-path     inet:uri
     +--rw default-schema-set?   -> /schema-selection/schema-sets/name {default-
schema-set}?

  rpcs:
    +---x select-schema-sets
       +---w input
          +---w schema-sets* [schema-set]
             +---w schema-set    -> /schema-selection/schema-sets/name
```

10. YANG Module

The YANG module definition for the module described in the previous sections.

```
<CODE BEGINS> file "ietf-schema-version-selection@2019-10-31.yang"
module ietf-schema-version-selection {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-schema-version-selection";
  prefix "ver-sel";

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types.";
  }
  import ietf-datastores {
    prefix ds;
    reference
```

```
      "RFC 8342: Network Management Datastore Architecture (NMDA)";
  }
  import ietf-yang-library {
    prefix yanglib;
    reference "RFC 8525: YANG Library";
  }
  import ietf-yl-packages {
    prefix pkg;
    reference "draft-rwilton-netmod-yang-packages-02";
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";

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

     Author:   Reshad Rahman
               <mailto:rrahman@cisco.com>

     Author:   Rob Wilton
               <mailto:rwilton@cisco.com>";

  description
    "This module provide a data model to advertise and allow the
     selection of schema versions by clients.

     Copyright (c) 2019 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; see
     the RFC itself for full legal notices.";

  // RFC Ed.: update the date below with the date of RFC publication
  // and remove this note.
  // RFC Ed.: replace XXXX with actual RFC number and remove this
  // note.
  revision 2019-10-31 {
    description
      "Initial revision";
```

```
    reference
      "RFC XXXX: YANG Schema Version Selection";
  }

  /*
   * Features
   */

  feature "default-schema-set" {
    description
      "Feature that allows clients to choose the default schema set
       to be used for clients.

       Implementations may choose to only support this feature in
       <operational> to report the default-schema-set without
       allowing it to be configured.";
  }

  feature "secondary-schema-set" {
    description
      "Feature to choose if secondary schema sets may be configured by
       clients.

       Implementations may choose to only support this feature in
       <operational> to report secondary schema sets without
       allowing them to be configured.";
  }

  container schema-selection {
      description
        "YANG schema version selection";

    list schema-sets {
      key "name";

      description
        "All schema-sets that are available for selection by clients.";

      leaf name {
        type "string" {
          length "1..255";
        }
        description
          "The server assigned name of the schema-set.

           This should include the schema family, and appropriate
           versioning or release information";
      }
```

```
      list datastores {
        config false;
        key "datastore";

        description
          "The list of datastores supported for this schema set";

        leaf datastore {
          type ds:datastore-ref;
          description
            "The datastore that this datastore schema is associated
             with";
          reference
            "RFC 8342: Network Management Datastore Architecture
             (NMDA)";
        }

        list packages {
          key "package";
          description
            "YANG packages associated with this datastore schema";

          leaf package {
            type leafref {
              path "/yanglib:yang-library/pkg:package/pkg:name";
            }
            description
              "The name of the YANG package this schema relates to";
          }
          leaf version {
            type leafref {
              path '/yanglib:yang-library/'
               + 'pkg:package[pkg:name = current()/../package]/'
               + 'pkg:version';
            }

            description
              "The version of the YANG package this schema relates to";
          }
        }
      }
    }


    container restconf {
      if-feature "secondary-schema-set";

      description
```

```
        "RESTCONF protocol settings for schema sets";

      list schema-sets {
        key "schema-set";
        unique "root-path";
        description "The schema-sets accessed via RESTCONF";

        leaf schema-set {
          type leafref {
            path '/schema-selection/schema-sets/name';
          }
          description "A schema-set being used by RESTCONF";
        }
        leaf root-path {
          type inet:uri;
          mandatory true;
          description
            "The root path to use to access the RESTCONF
            protocol instance for this schema-set";
          reference
            "RFC8040";
        }
      }
    }

    leaf default-schema-set {
      if-feature "default-schema-set";
      type leafref {
        path '/schema-selection/schema-sets/name';
      }
      description
        "Specifies the default schema-set used by this device.  This
         is the set of datastore schema that is used if a client
         connects using the standard URLs";
    }
  }

  /*
   * RPCs
   */
   rpc select-schema-sets {
     description
       "This RPC allows a NETCONF client to select which schema-sets, among the
        ones advertised by the server, the clients wants to use for this session
";

     input {
       list schema-sets {
         key "schema-set";
```

```
          leaf schema-set {
            type leafref {
              path '/schema-selection/schema-sets/name';
            }
          }
        }
      }
    }
}
<CODE ENDS>
```

11. Security Considerations

To be defined.

12. IANA Considerations

TODO - Add registrations for YANG modules defined in this document.

This document registers a URI.

12.1. NETCONF Capability URNs

This document registers a URI in the IETF XML registry [RFC3688]. The IANA registry "Network Configuration Protocol (NETCONF) Capability URNs" needs to be updated to include the following capability.

Index Capability Identifier
:schema-sets urn:ietf:params:netconf:capability:schema-sets:1.0

13. Open Questions/Issues

All issues, along with the draft text, are currently being tracked at: https://github.com/netmod-wg/yang-ver-dt/labels/version-selection-solution

14. Acknowledgements

The ideas that formed this draft are based on discussions with the YANG versioning design team, and other members of the NETMOD WG.

15. References

15.1. Normative References

[I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-07 (work in progress), October 2018.

[I-D.ietf-netmod-module-tags]
Hopps, C., Berger, L., and D. Bogdanovic, "YANG Module Tags", draft-ietf-netmod-module-tags-09 (work in progress), September 2019.

[I-D.ietf-netmod-yang-instance-file-format]
Lengyel, B. and B. Claise, "YANG Instance Data File Format", draft-ietf-netmod-yang-instance-file-format-04 (work in progress), August 2019.

[I-D.rwilton-netmod-yang-packages]
Wilton, R., "YANG Packages", draft-rwilton-netmod-yang-packages-02 (work in progress), October 2019.

[I-D.verdt-netmod-yang-module-versioning]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "Updated YANG Module Revision Handling", draft-verdt-netmod-yang-module-versioning-01 (work in progress), October 2019.

[I-D.verdt-netmod-yang-semver]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "YANG Semantic Versioning", draft-verdt-netmod-yang-semver-01 (work in progress), October 2019.

[I-D.verdt-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", draft-verdt-netmod-yang-versioning-reqs-02 (work in progress), November 2018.

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[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, <https://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, <https://www.rfc-editor.org/info/rfc6242>.

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

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

15.2. Informative References

[I-D.bierman-netmod-yang-package]
Bierman, A., "The YANG Package Statement", draft-bierman-netmod-yang-package-00 (work in progress), July 2015.

[I-D.ietf-netmod-artwork-folding]
Watsen, K., Farrel, A., and Q. WU, "Handling Long Lines in Inclusions in Internet-Drafts and RFCs", draft-ietf-netmod-artwork-folding-10 (work in progress), September 2019.

[RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <https://www.rfc-editor.org/info/rfc8199>.

Authors' Addresses

Robert Wilton
Cisco Systems, Inc.

Email: rwilton@cisco.com

Reshad Rahman
Cisco Systems, Inc.

Email: rrahman@cisco.com

Joe Clarke
Cisco Systems, Inc.

Email: jclarke@cisco.com

NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2020

M. Wang
Q. Wu
Huawei
C. Xie
China Telecom
I. Bryskin
Individual
X. Liu
Volta Networks
A. Clemm
Futurewei
H. Birkholz
Fraunhofer SIT
T. Zhou
Huawei

October 31, 2019

A YANG Data model for ECA Policy Management
draft-wwx-netmod-event-yang-04

Abstract

RFC8328 defines a policy-based management framework that allow definition of a data model to be used to represent high-level, possibly network-wide policies. Policy discussed in RFC8328 are classified into imperative policy and declarative policy, ECA policy is an typical example of imperative policy. This document defines an YANG data model for the ECA policy management. The ECA policy YANG provides the ability for the network management function (within a controller, an orchestrator, or a network element) to control the configuration and monitor state change on the network element and take simple and instant action when a trigger condition on the system state is met.

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 https://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 3, 2020.

Copyright Notice

Copyright (c) 2019 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 (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
2. Conventions used in this document 4
2.1. Terminology 4
2.2. Tree Diagrams 4
3. Objectives 4
4. Relationship to YANG Push 5
5. Model Overview 6
6. EVENT TRIGGER YANG Module 12
7. EVENT YANG Module 17
8. Security Considerations 23
9. IANA Considerations 24
10. Acknowledges 24
11. Contributors 25
12. Normative References 25
Appendix A. Usage Example of ECA model Working with YANG PUSH . 26
Appendix B. Usage Example of Reusing Trigger-Grouping 29
Appendix C. Changes between Revisions 30
Authors' Addresses 31

1. Introduction

Network management consists of using one or multiple device-, technology-, service specific policies to influence management behavior within the system and make sure policies are enforced or executed correctly.

[RFC8328] defines a policy-based management framework that allow definition of a data model to be used to represent high-level, possibly network-wide policies. Policies discussed in [RFC8328] are classified into imperative policy and declarative policy. Declarative policy specifies the goals to be achieved but not how to achieve those goals while imperative policy specifies when Events are triggered and what actions must be performed on the occurrence of an event. ECA policy is a typical example of imperative policy.

Event-driven management of states of managed objects across a wide range of devices can be used to monitor state changes of managed objects or resource and automatic trigger of rules in response to events so as to better service assurance for customers and to provide rapid autonomic response that can exhibit self-management properties including self-configuration, self-healing, self-optimization, and self-protection. Following are some of the use-cases where such ECA Policy can be used:

- To filter out of objects underneath a requested a subtree, the subscriber may use YANG Push smart filter to request the network server to monitor specific network management data objects and send updates only when the value falls within a certain range.

- To filter out of objects underneath a requested a subtree, the subscriber may use YANG Push smart filter to request the network server to monitor specific network management data objects and send updates only when the value exceeds a certain threshold for the first time but not again until the threshold is cleared.

- To provide rapid autonomic response that can exhibit self-management properties, the management system delegate event response behaviors (e.g., auto-recover from network failure) to the network device so that the network can react to network change as quickly as the event is detected. The event response behaviors delegation can be done using ECA policy,e.g., to preconfigure protection/ restoration capability on the network device.

- To perform troubleshoot failures (i.e., fault verification and localization) and provide root cause analysis, the management system monitoring specific network management data objects may request the network device to export state information of a set of managed data objects when the value of monitored data object exceeds a certain threshold.

This document defines a ECA Policy management YANG data model. The ECA Policy YANG provides the ability for the network management function (within a controller, an orchestrator, or a network element) to control the configurations and monitor state parameters on the

network element and take simple and instant action when a trigger condition on the system state is met.

The data model in this document is designed to be compliant with the Network Management Datastore Architecture (NMDA) [RFC8342].

2. Conventions used in this document

2.1. Terminology

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

This document uses the following terms:

Error A deviation of a system from normal operation [RFC3877].

Fault Lasting error or warning condition [RFC3877].

Event Something that happens which may be of interest or trigger the invocation of the rule. A fault, an alarm, a change in network state, network security threat, hardware malfunction, buffer untilization crossing a threshold, network connection setup, an external input to the system, for example [RFC3877].

2.2. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

3. Objectives

This section describes some of the design objectives for the ECA Policy management Data Model:

- Clear and precise identification of Event types in the ECA Policy.
- Clear and precise identification of managed object in the ECA Policy.
- Allow nested ECA policy,e.g, one event to be able to call another nested event.

- Allow the NETCONF server send updates only when the value falls within a certain range.

- Allow the NETCONF server send updates only when the value exceeds a certain threshold for the first time but not again until the threshold is cleared.

- Provide rapid autonomic response in the network device that can exhibit self-management properties including self-configuration, self-healing, self-optimization, and self-protection.

4. Relationship to YANG Push

YANG-push mechanism provides a subscription service for updates from a datastore. And it supports two types of subscriptions which are distinguished by how updates are triggered: periodic and on-change.

The On-change Push allow receivers to receive updates whenever changes to target managed objects occur. This document specifies a mechanism that provides three trigger conditions:

- Existence: When a specific managed object appears,disappear or object change, the trigger fires, e.g. reserved ports are configured.

- Boolean: The user can set the type of boolean operator (e.g. unequal, equal, less, less-or-equal, greater, greater-or-equal, etc) and preconfigured threshold value (e.g. Pre-configured threshold). If the value of a managed object meet Boolean conditions, the trigger fires, e.g., when the boolean operator type is 'less', the trigger will be fired if the value of managed object is less than the pre-configured Boolean value.

- Threshold: The user can set the rising threshold,the falling threshold, the delta rising threshold, the delta falling threshold. A threshold test regularly compares the value of the monitored object with the threshold values, e.g., an event is triggered if the value of the monitored object is greater than or equal to the rising threshold or an event is triggered if the difference between the current measurement value and the previous measurement value is smaller than or equal to the delta falling threshold.

In these three trigger conditions, existence with type set to object change is similar to on Push change.

In addtion, the model defined in this document provides a method for closed loop network management automation which allows automatic

trigger of rules in response to events so as to better service assurance for customers and to provide rapid autonomic response that can exhibit self-management properties including self-configuration, self-healing, self-optimization, and self-protection. The details of the usage example is described in Appendix A.

5. Model Overview

The YANG data model for the ECA Policy management has been split into two modules:

- The ietf-event-trigger.yang module defines a set of groupings for a generic trigger. It is intended that these groupings will be used by the policy based event management model or other models that require the trigger conditions. In this model, three trigger conditions are defined under the "test" choice node:
  - Existence: An existence test monitors and manages the absence, presence, and change of a data object, for example, interface status. When a monitored object is specified, the system reads the value of the monitored object regularly.
    - If the test type is Absent, the system triggers a network event and takes the specified action when the monitored object disappears.
    - If the test type is Present, the system triggers a network event and takes the specified action when the monitored object appears.
    - If the test type is Changed, the system triggers a network event and takes the specified action when the value of the monitored object changes.
  - Boolean: A Boolean test compares the value of the monitored object with the reference value and takes action according to the comparison result. The comparison types include unequal, equal, less, lessorequal, greater, and greaterorequal. For example, if the comparison type is equal, an event is triggered when the value of the monitored object equals the reference value. The event will not be triggered again until the value becomes unequal and comes back to equal.
  - Threshold: A Threshold trigger condition regularly compares compares the value of the monitored object with the threshold values.

+ A rising network event is triggered if the value of the monitored object is greater than or equal to the rising threshold.

+ A falling network event is triggered if the value of the monitored object is smaller than or equal to the falling threshold.

+ A rising network event is triggered if the difference between the current measurement value and the previous measurement value is greater than or equal to the delta rising threshold.

+ A falling network event is triggered if the difference between the current measurement value and the previous measurement value is smaller than or equal to the delta falling threshold.

+ A falling network event is triggered if the values of the monitored object, the rising threshold, and the falling threshold are the same.

+ A falling network event is triggered if the delta rising threshold, the delta falling threshold, and the difference between the current sampled value and the previous sampled value is the same.

If the value of the monitored object crosses a threshold multiple times in succession, the managed device triggers a network event only for the first crossing.

Editor-note: Three trigger conditions defined this document align with Complex condition and Simple condition defined in RFC3460? Are they sufficient to model ECA condition?

o The ietf-event.yang module defines four lists: trigger, target, event, and action. Triggers define the targets meeting some conditions that lead to events. Events trigger corresponding actions:

  * Each trigger can be seen as a logical test that, if satisfied or evaluated to be true, cause the action to be carried out. The ietf-event.yang module uses groupings defined in ietf-event-trigger.yang to present the trigger attributes.

  * The target list defines managed objects that can be added to logging or be set to a new value on the trigger, the trigger test type, or the event that resulted in the actions. The

target is also refered to as policy variable defined in [RFC3460].

* The event list defines what happens when an event is triggered, i.e., trigger the corresponding action, e.g., addding a logging (i.e. Recording the triggered event), setting a value to the managed object or both. The group-id can be used to group a set of event that can be executed together,e.g., deliver a service or provide service assurance.

* Nested-event are supported by allowing one event's trigger to reference other event's definitions using the call-event configuration. Called events apply their triggers and actions before returning to the calling event's trigger and resuming evaluation. If the called event is triggered, then it returns an effective boolean true value to the calling event. For the calling event, this is equivalent to a condition statement evaluating to a true value and evaluation of the event continues.

* The action list consists of updates or invocations on local managed object attributes and defines a set of actions which will be performed (e.g. logging, set value, etc) when the corresponding event is triggered. The value to be set can use many variations on rule structure.

The following tree diagrams [RFC8340] provide an overview of the data model for "ietf-event-trigger" module and the "ietf-event" module.

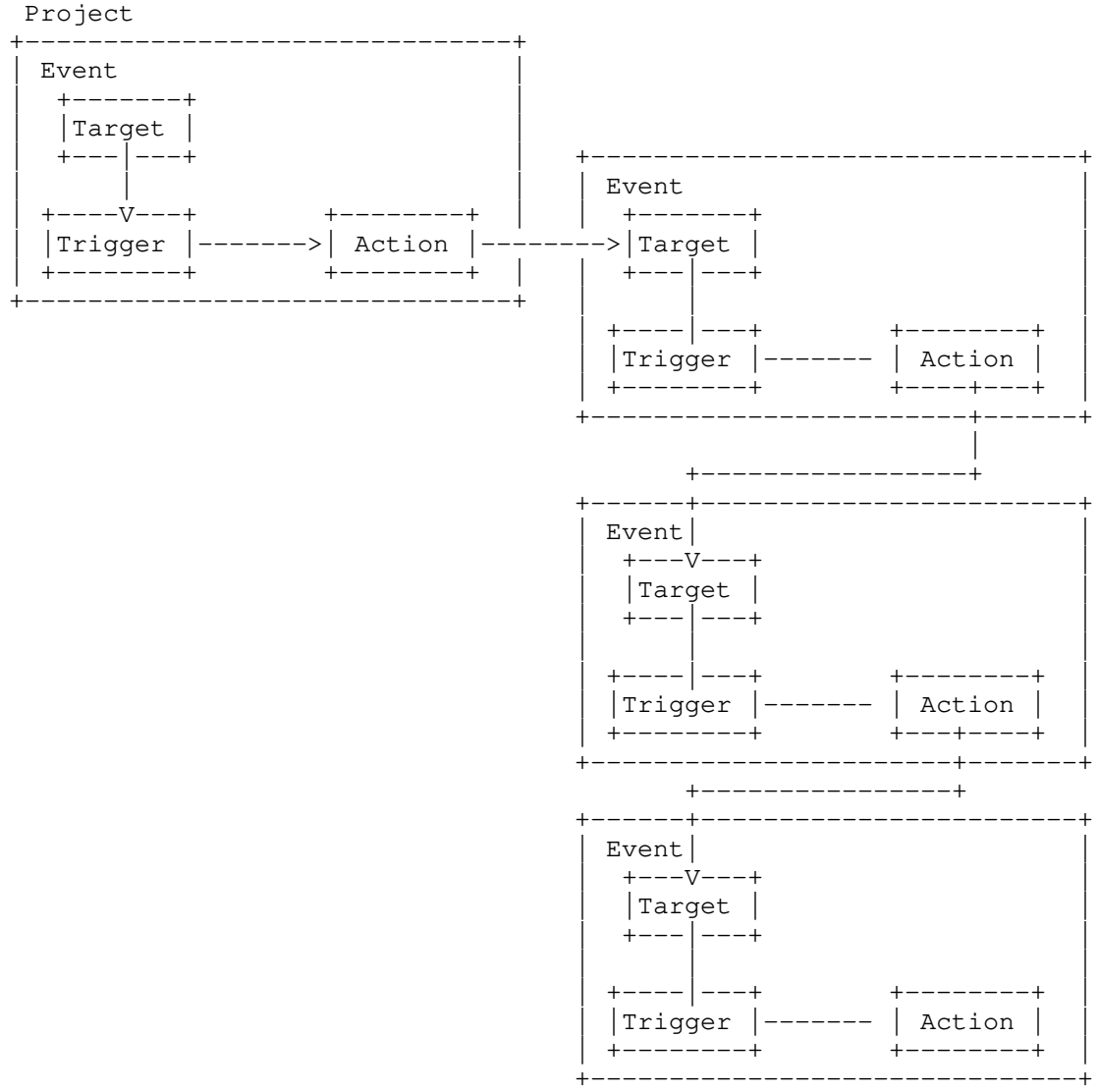
```
module: ietf-event-trigger
  grouping existences-trigger
    +-- existences
       +-- test-type?   enumeration
       +-- target*      target
  grouping boolean-trigger
    +-- boolean
       +-- operator?   operator
       +-- value?      match-value
       +-- target*     target
  grouping threshold-trigger
    +-- threshold
       +-- rising-value?           match-value
       +-- rising-target*          target
       +-- falling-value?          match-value
       +-- falling-target*         target
       +-- delta-rising-value?     match-value
       +-- delta-rising-target*    target
       +-- delta-falling-value?    match-value
```

```
         +-- delta-falling-target*   target
         +-- startup?                enumeration
     grouping trigger-grouping
       +-- (test)?
          +--:(existences)
          |  +-- existences
          |     +-- test-type?   enumeration
          |     +-- target*      target
          +--:(boolean)
          |  +-- boolean
          |     +-- operator?   operator
          |     +-- value?      match-value
          |     +-- target*     target
          +--:(threshold)
             +-- threshold
                +-- rising-value?           match-value
                +-- rising-target*          target
                +-- falling-value?          match-value
                +-- falling-target*         target
                +-- delta-rising-value?     match-value
                +-- delta-rising-target*    target
                +-- delta-falling-value?    match-value
                +-- delta-falling-target*   target
                +-- startup?                enumeration

   module: ietf-event
     +--rw events
        +--rw event* [event-name type]
           +--rw event-name           string
           +--rw type                 identityref
           +--rw event-description?   string
           +--rw group-id?            group-type
           +--rw target*              trig:target
           +--rw clear?               boolean
           +--rw trigger* [name]
           |  +--rw name                     string
           |  +--rw trigger-description?   string
           |  +--rw call-event?            -> ../../event-name
           |  +--rw frequency
           |  |  +--rw type?         identityref
           |  |  +--rw periodic
           |  |  |  +--rw interval    uint32
           |  |  |  +--rw start?      yang:date-and-time
           |  |  |  +--rw end?        yang:date-and-time
           |  |  +--rw scheduling
           |  |     +--rw month*          string
           |  |     +--rw day-of-month*   uint8
           |  |     +--rw day-of-week*    uint8
```

```
|  |     +--rw hour*           uint8
|  |     +--rw minute*         uint8
|  |     +--rw second*         uint8
|  |     +--rw start?          yang:date-and-time
|  |     +--rw end?            yang:date-and-time
|  +--rw (test)?
|     +--:(existences)
|     |  +--rw existences
|     |     +--rw target*   target
|     +--:(boolean)
|     |  +--rw boolean
|     |     +--rw operator?   operator
|     |     +--rw value?      match-value
|     |     +--rw target*     target
|     +--:(threshold)
|        +--rw threshold
|           +--rw rising-value?           match-value
|           +--rw rising-target*          target
|           +--rw falling-value?          match-value
|           +--rw falling-target*         target
|           +--rw delta-rising-value?     match-value
|           +--rw delta-rising-target*    target
|           +--rw delta-falling-value?    match-value
|           +--rw delta-falling-target*   target
|           +--rw startup?                enumeration
+--rw actions
   +--rw action* [name]
      +--rw name       string
      +--rw set
      |  +--rw target?   trig:target
      |  +--rw value?    <anydata>
      +--rw logging
         +--rw type?   logging-type
```

The relation between Event, Trigger, Target and Action is described as follows:



One event may trigger another event,i.e., the action output in the first event can be input to target in the second event and a set of events can be grouped together and executed in a coordinated manner, but if it does not trigger another event, the relation between two events should be ignored.

6. EVENT TRIGGER YANG Module

```
<CODE BEGINS> file "ietf-event-trigger@2019-10-28.yang"
module ietf-event-trigger {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-event-trigger";
  prefix trig;

  import ietf-yang-types {
    prefix yang;
  }
  organization
    "IETF xxx Working Group";
  contact
    "Zitao Wang: wangzitao@huawei.com
     Qin Wu: bill.wu@huawei.com";
  description
    "This module defines a reusable grouping for event trigger.";

  revision 2019-10-28 {
    description
      "Initial revision.";
    reference
      "foo";
  }

  typedef match-value {
    type union {
      type yang:xpath1.0;
      type yang:object-identifier;
      type string;
    }
    description
      "This type is used to match resources of type 'target'.
       Since the type 'target' is a union of different types,
       the 'match-value' type is also a union of corresponding
       types.";
  }

  typedef target {
    type union {
      type instance-identifier;
      type yang:object-identifier;
      type yang:uuid;
      type string;
    }
    description
      "If the target is modelled in YANG, this type will
```

```
       be an instance-identifier.
       If the target is an SNMP object, the type will be an
       object-identifier.
       If the target is anything else, for example a distinguished
       name or a CIM path, this type will be a string.
       If the target is identified by a UUID use the uuid
       type.
       If the server supports several models, the presedence should
       be in the order as given in the union definition.";
  }

  typedef operator {
    type enumeration {
      enum unequal {
        description
          "Indicates that the comparision type is unequal to.";
      }
      enum equal {
        description
          "Indicates that the comparision type is equal to.";
      }
      enum less {
        description
          "Indicates that the comparision type is less than.";
      }
      enum less-or-equal {
        description
          "Indicates that the comparision type is less than
           or equal to.";
      }
      enum greater {
        description
          "Indicates that the comparision type is greater than.";
      }
      enum greater-or-equal {
        description
          "Indicates that the comparision type is greater than
           or equal to.";
      }
    }
    description
      "definition of the operator";
  }

  grouping existences-trigger {
    description
      "A grouping that provides existence trigger";
    container existences {
```

```
      leaf test-type {
        type enumeration {
          enum absent {
            description
              "If the test type is Absent, the system triggers an event
               and takes the specified action when the monitored object disappea
rs.";
          }
          enum present {
            description
              "If the test type is Present, the system triggers an event
               and takes the specified action when the monitored object appears.
";
          }
          enum changed {
            description
              "If the test type is Changed, the system triggers an alarm event a
nd takes
               the specified action when the value of the monitored object chang
es.";
          }
        }
       description "test type.";
      }
      leaf-list target {
        type target;
        description
          "List for target objects";
      }
      description
        "Container for existence";
    }
  }

  grouping boolean-trigger {
    description
      "A grouping that provides boolean trigger";
    container boolean {
      leaf operator {
        type operator;
        description
          "Comparison type.";
      }
      leaf value {
        type match-value;
        description
          "Compartion value which is static threshold value.";
      }
      leaf-list target {
        type target;
        description
          "List for target management objects.";
```

```
      }
      description
        "Container for boolean test.";
    }
  }

  grouping threshold-trigger {
    description
      "A grouping that provides threshold trigger";
    container threshold {
      leaf rising-value {
        type match-value;
        description
          "Sets the rising threshold to the specified value,
           when the current sampled value is greater than or equal to
           this threshold, and the value at the last sampling interval
           was less than this threshold, the event is triggered. ";
      }
      leaf-list rising-target {
        type target;
        description
          "List for target objects.";
      }
      leaf falling-value {
        type match-value;
        description
          "Sets the falling threshold to the specified value.";
      }
      leaf-list falling-target {
        type target;
        description
          "List for target objects.";
      }
      leaf delta-rising-value {
        type match-value;
        description
          "Sets the delta rising threshold to the specified value.";
      }
      leaf-list delta-rising-target {
        type target;
        description
          "List for target objects.";
      }
      leaf delta-falling-value {
        type match-value;
        description
          "Sets the delta falling threshold to the specified value.";
      }
```

```
      leaf-list delta-falling-target {
        type target;
        description
          "List for target objects.";
      }
      leaf startup {
        type enumeration {
          enum rising {
            description
              "If the first sample after this
               managed object becomes active is greater than or equal
                to 'rising-value' and the 'startup' is equal to
                'rising' then one threshold rising event is
               triggered for that managed object.";
          }
          enum falling {
            description
              "If the first sample after this managed object becomes
               active is less than or equal to 'falling-value' and
               the 'startup' is equal to 'falling' then one
               threshold falling event is triggered for that managed
               object.";
          }
          enum rising-or-falling {
            description
              "That event may be triggered when the
               'startup' is equal to 'rising-or-falling'.
               'rising-or-falling' indicate the state value of the
                managed object may less than or greater than the
               specified thrshold value.";
          }
        }
        description
          "Startup setting.";
      }
      description
        "Container for the threshold trigger condition.
         Note that the threshold here  may change over time
         or the state value changes in either ascend order
         or descend order.";
    }
  }

  grouping trigger-grouping {
    description
      "A grouping that provides event trigger.";
    choice test {
      description
```

```
        "Choice test";
       case existences {
         uses existences-trigger;
       }
       case boolean {
         uses boolean-trigger;
       }
       case threshold {
         uses threshold-trigger;
       }
     }
   }
}
<CODE ENDS>
```

7. EVENT YANG Module

```
   <CODE BEGINS> file "ietf-event@2019-10-28.yang"

   module ietf-event {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-event";
     prefix evt;

     import ietf-yang-types {
       prefix yang;
     }
     import ietf-event-trigger {
       prefix trig;
     }

     organization
       "IETF xxx Working Group";
     contact
       "Zitao Wang: wangzitao@huawei.com
        Qin Wu: bill.wu@huawei.com";
     description
       "This module defines a model for the service topology.";

     revision 2019-10-28 {
       description
         "Initial revision.";
       reference
         "foo";
     }

     identity event-type {
       description
```

```
        "Base identity for event type";
    }

    identity frequency {
      description
        "Base identity for frequency";
    }

    identity periodic {
      base frequency;
      description
        "Identity for periodic trigger";
    }

    identity scheduling {
      base frequency;
      description
        "Identity for scheduling trigger";
    }

    identity logging {
      description
        "Base identity for logging action";
    }

    identity logging-notification {
      base logging;
      description
        "Logging for event notification";
    }

    identity logging-set {
      base logging;
      description
        "Logging for reset values";
    }

    typedef logging-type {
      type identityref {
        base logging;
      }
      description
        "Logging types";
    }

    typedef group-type {
      type string;
      description
```

```
        "Group type";
    }

    grouping start-end-grouping {
      description
        "A grouping that provides start and end times for
         Event objects.";
      leaf start {
        type yang:date-and-time;
        description
          "The date and time when the Event object
           starts to create triggers.";
      }
      leaf end {
        type yang:date-and-time;
        description
          "The date and time when the Event object
           stops to create triggers.
           It is generally a good idea to always configure
           an end time and to refresh the end time as needed
           to ensure that agents that lose connectivity to
           their Controller do not continue executing Schedules
           forever.";
      }
    }

    container events {
      list event {
        key "event-name type";
        leaf event-name {
          type string;
          description
            "Event name";
        }
        leaf type {
          type identityref {
            base event-type;
          }
          description
            "Type of event";
        }
        leaf event-description {
          type string;
          description
            "Event description";
        }
        leaf group-id {
          type group-type;
```

```
          description
            "Group Identifier";
        }
        leaf-list target {
          type trig:target;
          description
            "targeted objects";
        }
        leaf clear {
          type boolean;
          default "false";
          description
            "A flag indicate whether the event be closed";
        }
        list trigger {
          key "name";
          leaf name {
            type string;
            description
              "Trigger name";
          }
          leaf trigger-description {
            type string;
            description
              "Trigger description";
          }
          leaf call-event {
            type leafref {
              path "../../event-name";
            }
            description
              "This leaf call sub-event.";
          }
          container frequency {
            leaf type {
              type identityref {
                base frequency;
              }
              description
                "Type of trigger frequency";
            }
            container periodic {
              when "derived-from-or-self(../type, 'periodic')";
              description
                "A periodic timing object triggers periodically
                 according to a regular interval.";
              leaf interval {
                type uint32 {
```

```
            range "1..max";
          }
          units "seconds";
          mandatory true;
          description
            "The number of seconds between two triggers
             generated by this periodic timing object.";
        }
        uses start-end-grouping;
      }
      container scheduling {
        when "derived-from-or-self(../type, 'scheduling')";
        description
          "A scheduling timing object triggers.";
        leaf-list month {
          type string;
          description
            "A set of months at which this scheduling timing
             will trigger.";
        }
        leaf-list day-of-month {
          type uint8 {
            range "0..59";
          }
          description
            "A set of days of the month at which this
             scheduling timing will trigger.";
        }
        leaf-list day-of-week {
          type uint8 {
            range "0..59";
          }
          description
            "A set of weekdays at which this scheduling timing
             will trigger.";
        }
        leaf-list hour {
          type uint8 {
            range "0..59";
          }
          description
            "A set of hours at which the scheduling timing will
             trigger.";
        }
        leaf-list minute {
          type uint8 {
            range "0..59";
          }
```

```
              description
                "A set of minutes at which this scheduling timing
                 will trigger.";
            }
            leaf-list second {
              type uint8 {
                range "0..59";
              }
              description
                "A set of seconds at which this calendar timing
                 will trigger.";
            }
            uses start-end-grouping;
          }
          description
            "Container for frequency";
        }
        uses trig:trigger-grouping;
        description
          "List for trigger";
      }
      container actions {
        list action {
          key "name";
          leaf name {
            type string;
            description
              "Action Name";
          }
          container set {
            leaf target {
              type trig:target;
              description
                "The target objects";
            }
            anydata value {
              description
                "Inline set content.";
            }
            description
              "Set a value to the target";
          }
          container logging {
            leaf type {
              type logging-type;
              description
                "Specifies the log action";
            }
```

```
              description
                "Specifies the log action";
            }
            description
              "List for actions";
          }
          description
            "Container for Actions";
        }
        description
          "List for Events";
      }
      description
        "YANG data module for defining event triggers and actions for
         network management purposes";
    }
  }
```

<CODE ENDS>

8. Security Considerations

The YANG modules defined in this document MAY be accessed via the RESTCONF protocol [RFC8040] or NETCONF protocol ([RFC6241]). The lowest RESTCONF or NETCONF layer requires that the transport-layer protocol provides both data integrity and confidentiality, see Section 2 in [RFC8040] and [RFC6241]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH)[RFC6242] . The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

- /events/event/event-name

- /events/event/target

- /events/actions/target

- /events/event/trigger/name

9. IANA Considerations

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested to be made:

```
---------------------------------------------------------------------
   URI: urn:ietf:params:xml:ns:yang:ietf-event-trigger
   Registrant Contact: The IESG.
   XML: N/A, the requested URI is an XML namespace.

   URI: urn:ietf:params:xml:ns:yang:ietf-event
   Registrant Contact: The IESG.
   XML: N/A, the requested URI is an XML namespace.
---------------------------------------------------------------------
```

This document registers two YANG modules in the YANG Module Names registry [RFC6020].

```
---------------------------------------------------------------------
   Name:         ietf-event-trigger
   Namespace:    urn:ietf:params:xml:ns:yang:ietf-event-trigger
   Prefix:       trig
   Reference:    RFC xxxx

   Name:         ietf-event
   Namespace:    urn:ietf:params:xml:ns:yang:ietf-event
   Prefix:       evt
   Reference:    RFC xxxx
---------------------------------------------------------------------
```

10. Acknowledges

This work has benefited from the discussions of ECA Policy over the years. In particular, the SUPA project [https://datatracker.ietf.org/wg/supa/about/] provided approaches to express high-level, possibly network-wide policies to a network management function (within a controller, an orchestrator, or a network element).

Igor Bryskin, Xufeng Liu, Alexander Clemm, Henk Birkholz, Tianran Zhou contributed to an earlier version of [GNCA]. We would like to thank the authors of that document on event response behaviors

delegation for material that assisted in thinking that helped improve this document.

11. Contributors

Nicola Sambo
Scuola Superiore Sant'Anna
Via Moruzzi 1
Pisa 56124
Italy

Email: nicola.sambo@sssup.it

Giuseppe Fioccola
Huawei Technologies
Riesstrasse, 25
Munich 80992
Germany

Email: giuseppe.fioccola@huawei.com

12. Normative References

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

[RFC2981] Kavasseri, R., Ed., "Event MIB", RFC 2981, DOI 10.17487/RFC2981, October 2000, <https://www.rfc-editor.org/info/rfc2981>.

[RFC3460] Moore, B., Ed., "Policy Core Information Model (PCIM) Extensions", RFC 3460, DOI 10.17487/RFC3460, January 2003, <https://www.rfc-editor.org/info/rfc3460>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6370] Bocci, M., Swallow, G., and E. Gray, "MPLS Transport Profile (MPLS-TP) Identifiers", RFC 6370, DOI 10.17487/RFC6370, September 2011, <https://www.rfc-editor.org/info/rfc6370>.

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

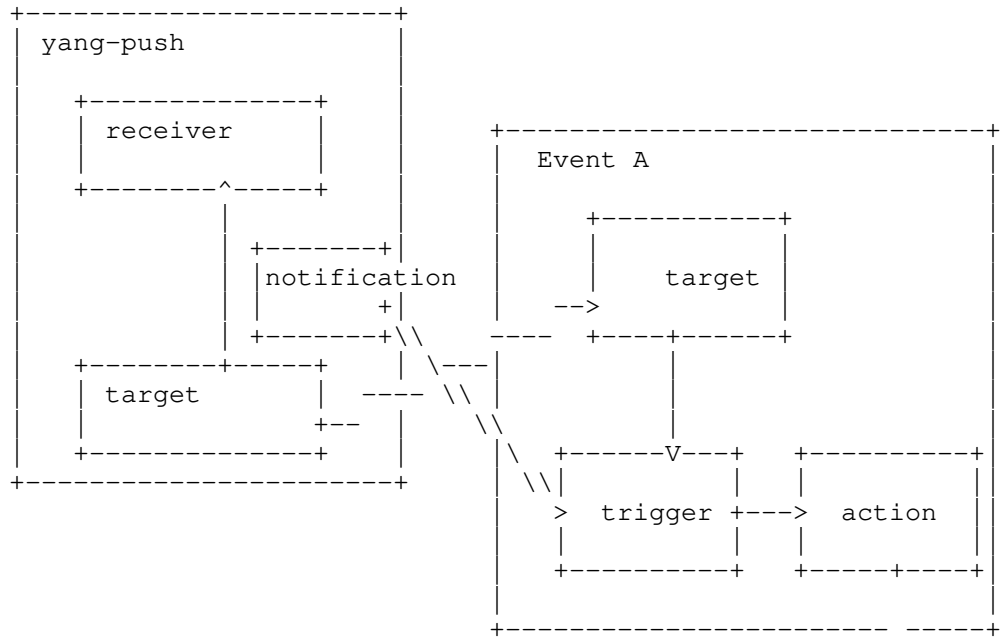
[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <https://www.rfc-editor.org/info/rfc7952>.

[RFC8328] Liu, W., Xie, C., Strassner, J., Karagiannis, G., Klyus, M., Bi, J., Cheng, Y., and D. Zhang, "Policy-Based Management Framework for the Simplified Use of Policy Abstractions (SUPA)", RFC 8328, DOI 10.17487/RFC8328, March 2018, <https://www.rfc-editor.org/info/rfc8328>.

Appendix A. Usage Example of ECA model Working with YANG PUSH

The relation between Event and YANG PUSH is described as follow: YANG Push Notification may trigger one event, i.e. one trigger conditions of the Event A can be set to "receiver received a yang push notification", and it can associates with other conditions of the Event A. When these conditions are met, the event A is triggered.



For Example:

The receiver received a push-change-update notification and learned that the "oper-status" of interface[name='eth0'] changed.

The target of Event "interface-state-monitoring" is set to "/if:interfaces/if:interface[if:name='eth0']", the trigger list contains two conditions: 1)receiver received a push-change-update notification; 2) the value of "in-errors" of interface[name='eth0'] exceeded the pre-configured threshold. When these conditions are met, corresponding action will be performed, i.e. disable interface[name='eth0']. The XML examples are shown as below:

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
 <eventTime>2017-10-25T08:22:33.44Z</eventTime>
 <push-change-update
      xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
   <id>89</id>
   <datastore-changes>
     <yang-patch>
       <patch-id>0</patch-id>
       <edit>
         <edit-id>edit1</edit-id>
```

```
           <operation>replace</operation>
           <target>/ietf-interfaces:interfaces</target>
           <value>
             <interfaces
                  xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
               <interface>
                 <name>eth0</name>
                 <oper-status>up</oper-status>
               </interface>
             </interfaces>
           </value>
         </edit>
       </yang-patch>
     </datastore-changes>
   </push-change-update>
  </notification>

    <event>
     <event-name>interface-state-monitoring</event-name>
     <type>interface-exception</type>
     <target>/if:interfaces/if:interface[if:name='eth0']</target>
     <trigger>
      <name>state-push-change</name>
      <trigger-description>received yang push
\changed notification</trigger-description>
      <test>
       <existence>/yp:notification/yp:push-change-update/yp:id[id=89]\
/yp:datastore-changes/.../yp:target="/ietf-interfaces:interfaces='eth0'\
"</existence>
      </test>
     </trigger>
     <trigger>
      <name>evaluate-in-errors</name>
      <call-event>interface-state-chang</call-event>
      <trigger-description>evaluate the number of
       the packets that contained errors
       </trigger-description>
      <frequency>10m</frequency>
      <test>
       <boolean>
        <operator>greater-or-equal</operator>
        <value>100</value>
        <target>/if:interfaces/if:interface[if:name='eth0']\
/if:statistic/if:in-errors</target>
       </boolean>
      </test>
     </trigger>
     <action>
```

```
      <target>/if:interfaces/if:interface[if:name='eth0']</target>
      <value>
       <interfaces>
        <interface>
         <name>eth0</name>
         <enable>false</enable>
        </interface>
       </interfaces>
      </value>
    </action>
   </event>
  </events>
```

Appendix B. Usage Example of Reusing Trigger-Grouping

The "ietf-event-trigger.yang" module defines a set of groupings for a generic trigger. It is intended that these groupings can be reused by other models that require the trigger conditions, for example, in some subscription and notification cases, many applications do not require every update, only updates that are of certain interest. The following example describe how to reuse the "ietf-event-trigger" module to define the subscription and notification smarter filter.

```
    import ietf-subscribed-notifications {
     prefix sn;
   }
  import ietf-event-trigger {
   prefix trig;
  }

  augment "/sn:subscriptions/sn:subscription" {
    description "add the smart filter container";
    container smart-filter {
       description "It concludes filter configurations";
       uses trig:trigger-grouping;
    }
  }
```

The tree diagrams:

```
module: ietf-smart-filter
augment /sn:subscriptions/sn:subscription:
  +--rw smart-filter
     +--rw (test)?
        +--:(existences)
        |  +--rw existences
        |     +--rw target*   target
        +--:(boolean)
        |  +--rw boolean
        |     +--rw operator?   operator
        |     +--rw value?      match-value
        |     +--rw target*     target
        +--:(variation)
           +--rw variation
              +--rw rising-value?           match-value
              +--rw rising-target*          target
              +--rw falling-value?          match-value
              +--rw falling-target*         target
              +--rw delta-rising-value?     match-value
              +--rw delta-rising-target*    target
              +--rw delta-falling-value?    match-value
              +--rw delta-falling-target*   target
              +--rw startup?                enumeration
```

Appendix C. Changes between Revisions

v03 - v04

- Add text in introduction section to clarify the usage examples of ECA policy
- Update objective section to align with use cases.
- Clarify the relationship between target and policy variable.
- Change variation trigger condition back into threshold trigger condition and clarify the usage of three trigger conditions.
- Remove Event MIB related section.
- Add new coauthors.

v02 - v03

- Usage Example Update: add an usage example to introduce how to reuse the ietf-event-trigger module to define the subscription-notification smarter filter.

v01 - v02

- Introduce the group-id which allow group a set of events that can be executed together
- Change threshold trigger condition into variation trigger condition to further clarify the difference between boolean trigger condition and variation trigger condition.
- Module structure optimization.
- Usage Example Update.

v00 - v01

- Separate ietf-event-trigger.yang from Event management modeland ietf-event.yang and make it reusable in other YANG models.
- Clarify the difference between boolean trigger condition and threshold trigger condition.
- Change evt-smp-min and evt-smp-max into min-data-object and max-data-object in the data model.

Authors' Addresses

Michael Wang
Huawei Technologies,Co.,Ltd
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: wangzitao@huawei.com

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

Email: bill.wu@huawei.com

Chongfeng Xie
China Telecom

Email: xiechf@ctbri.com.cn

Igor Bryskin
Individual

Email: i_bryskin@yahoo.com

Xufeng Liu
Volta Networks

Email: xufeng.liu.ietf@gmail.com

Alexander Clemm
Futurewei

Email: ludwig@clemm.org

Henk Birkholz
Fraunhofer SIT

Email: henk.birkholz@sit.fraunhofer.de

Tianran Zhou
Huawei

Email: zhoutianran@huawei.com