OPSAWG
Internet-Draft
Intended status: Standards Track
Expires: 14 September 2023

C. Feng, Ed.
Huawei
T. Hu
CMCC
LM. Contreras
Telefonica I+D
Q. Wu
C. Yu
Huawei

13 March 2023

Incident Management for Network Services
draft-feng-opsawg-incident-management-00

Abstract

This document provides an architecture for the incident management system and related function interface requirements.

This document also defines a YANG module to support the incident lifecycle management. This YANG module is meant to provide a standard way to report, diagnose, and resolve incidents for the sake of enhanced network services.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Network performance management and fault management are used for monitoring and troubleshooting separately in networking infrastructures. Typically, metrics and alarms, transaction operations are monitored centrally and incident tickets are triggered accordingly. A YANG [RFC7950] data model for alarm management [RFC8632] defines a standard interface for alarm management.

A data model for Network and VPN Service Performance Monitoring [I-D.opsawg-yang-vpn-service-pm] defines a standard interface for performance management. In addition, distributed tracing mechanism defined in [W3C-Trace-Context] can also be used to follow, analyze and debug operations, such as configuration transactions, across multiple distributed systems.

However, alarm-centric solution described in [RFC8632] and performance-centric solution described in [I-D.opsawg-yang-vpn-service-pm], trace context-centric solution is based on a data source specific information and maintenance engineers' experience and fall short when keeping track of them separately in various different management systems, e.g., the frequency and quantity of alarms reported to Operating Support System (OSS) increased dramatically (in many cases multiple orders of magnitude) with the growth of service types and complexity, hard to aggregate in a single domain along with key performance metrics, various different events, notifications, overwhelm OSS platforms, result in low processing efficiency, inaccurate root cause identification and duplicated tickets.

Usually, the network modeling from device to different connection and service layers follows some existing standards. Once there are some failures happened on network devices, there could be some correlative alarms appeared on the upper layers. Theoretically, it is possible to compress a series of alarms into fewer incidents. The traditional working manner is also based on this correlation relationship. But the traditional working manner is time-consuming and labor-intensive which reduces efficiency. Additionally, it quite depends on the experience of maintenance engineers. Moreover, the investigation of some faults also depends on some other data like topology data or performance data. This complicates network troubleshooting, and the correlation of alarms and network services. Therefore, it is difficult to assess the impact of alarms on network services.

To address these challenges, an incident-centric solution is proposed, which also supports cross-domain or cross-layer root cause analysis and network troubleshooting. A network incident refers to an unexpected interruption of a network service, degradation of a network service quality, or sub-health of a network service while an alarm described in [RFC8632] represents an undesirable state in a resource that requires corrective actions. An alarm will always be reported when network resources are unexpected while an incident is reported only when network services are affected, e.g., symptoms (e.g.,CPU overloaded) at the device level defined in [I-D.opsawg-service-assurance-yang] or root cause alarms can be used to generate and report incidents when the network service is in sub-health state or gets degraded. An incident may be triggered by aggregation and analysis of multiple alarms or other network anomalies, for example, the protocols related to the interface fail to work properly due to the interface down, as a result, the network service becomes unavailable. An incident may also be raised through the analysis of some network performance metrics, for example, the delay or packet loss rate exceeds the threshold, causing degradation of the network service.

Artificial Intelligence (AI) and Machine Learning (ML) play a important role in the processing of large amounts of data with complex correlations. For example, Neural Network Algorithm or Hierarchy Aggregation Algorithm can be used to replace manual alarm correlation. Through online and offline learning, these algorithms can be continuously optimized to improve the efficiency of fault diagnosis.

This document defines the concepts, requirements, and architecture of incident management. The document also defines a YANG data model for incident lifecycle management, which improves troubleshooting efficiency, ensures network service quality, and improves network automation [RFC8969].

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.

The following terms are defined in [RFC8632] are not redefined here:

* alarm

The following terms are defined in this document:

Incident: An unexpected interruption of a network service, degradation of network service quality, or sub-health of a network service.

Incident management: Lifecycle management of incidents including incident identification, reporting, acknowledge, diagnosis, and resolution.

Incident management system: An entity which implements incident management. It include incident management agent and incident management client.

Incident management agent: An entity which provides some functions of incident management. For example, it can detect an incident, perform incident diagnosis, resolution and prediction,etc.

Incident management client: An entity which can manage incidents. For example, it can receive incident notifications, query the information of incidents, instruct the incident management agent to diagnose, resolve, etc.

3. Sample Use Cases

3.1. Incident-Based Trouble Tickets dispatching

Currently, the dispatching of trouble tickets is mostly based on dispatching alarms. Some operators' maintenance engineers monitor and identify alarms which could link to the same fault. Then they dispatch these alarms to the same trouble ticket, which is in low automation. If there are many alarms, then the human costs are increased accordingly.

Some operators preset whitelist and adopt some coarse granularity association rules for the alarm management. It seems to improve fault management automation. However, some trouble tickets could be missed if the filtering conditions are too tight. If the filtering conditions are too loose, multiple trouble tickets would be dispatched to the same fault.

It is hard to achieve a perfect balance between the automation and duplicated trouble tickets under the traditional working situations. However, with the help of incident management, massive alarms can be aggregated into a few incidents, multiple trouble tickets will be saved. At the same time, incident management can keep high accuracy and automation. This could be an answer to this pain point of traditional trouble ticket dispatching

3.2. Fault Locating

Currently, to accomplish fault isolation and locating work, maintenance experts need to combine topology data, service data with huge amount of alarm data to do the analysis. Sometimes they also require some cooperation from the construction engineers who work on site, to operate fixing attempts on devices and then further investigation the root cause is required.

For example, for a common cable interruption, maintenance experts need to analyze the root cause alarm from massive alarms, and then trace the root alarm to the faulty span segment by segment. Next, site engineers perform tests at the source station to locate the interruption and locate the faulty optical exchange station. Then travel to the located optical exchange station to replace or splice fibers. During the whole process, multiple people are needed inside and outside the site.

With the help of incident management, the system can automatically locate the faulty span, and eliminate the need for manual analysis. By cooperating with the integrated OTDR within the equipment, we can determine the target optical exchange station before site visits. Multiple site visits and time are saved.

3.3. Fault Labelling

Fiber cutover is a common maintenance scenario for Operators. During the cutover process, maintenance experts must identify affected devices based on the cutover object and their experience. They will give these devices a mark to remind other maintenance engineers that it is not necessary to dispatch trouble tickets before the ending of cutover.

However, depending on human experience, it is very likely to make some mistakes. For example, some devices are missing to mark and some devices are marked incorrectly. If the devices are missing to mark, some trouble tickets will be dispatched during cutover, which are not needed actually. If the devices are wrongly marked, some fault not related to this cutover will be missing.

With incident management, maintenance experts only need to mark the cutover objects and do not need to mark the devices that would be affected. Because of the alarm aggregation capabilities and knowing the relationship between root cause alarm and correlative alarm, the fault management system can automatically identify correlative alarms, without dispatching any trouble tickets to the affected devices.

3.4. Energy Conservation

Under the global trend of energy conservation, emission reduction and safety management, more and more enterprises have joined the energy conservation and emission reduction ranks and adopted measures to turn off the power after work during non-working hours, making due contributions to the green earth. However, this proactive power-off measure periodically generates a large number of alarms on the network, and the traditional Operation and Management system can not effectively identify such non-real faults caused by the enterprise users? operations. Operators need to manually identify and rectify faults based on expert experience, wasting a large number of human resources.

Incident management can intelligently identify faults caused by periodic power-off on the tenant side and directly identify faults. As a result, operators do not need to dispatch trouble tickets for such faults any more, this can help to reduce human resource costs.

4. Incident Management Architecture

```
               +----------------------+-------------------+
               |                                          |
               |            Incident Management Client    |
               |                                          |
               |                                          |
               +------------+---------+---------+---------+
                  ^         |         |         |
                  |Incident |Incident |Incident |Incident
                  |Report   |Ack      |Diagnose |Resolve
                  |         |         |         |
                  |         V         V         V
               +--+-------------------+---------+----------+
               |                                           |
               |                                           |
               |            Incident Management Agent      |
               |                                           |
               |                                           |
               |                                           |
               |                                           |
               +----------------------+-----+--+-----------+
                     ^       ^Abnormal         ^
                     |Alarm  |Operations       |Metrics
                     |Report |Report           |/Telemetry
                     |       |                 V
        +--------+-+-+-------+--------------++------------------+
        |                                                       |
        |                     Network                           |
        |                                                       |
        +------------------------------------+------------------+
```

Figure 1: Incident Management Architecture

Figure 1 illustrates the incident management architecture. Two key components for the incident management are incident management client and incident management agent.

Incident management agent can be deployed in network analytics platform, controllers or Orchestrators and provides functionalities such as incident detection, report, diagnosis, resolution, querying for incident lifecycle management.

Incident management client can be deployed in the network OSS or other business systems of operators and invokes the functionalities provided by incident management agent to meet the business requirements of fault management.

A typical workflow of incident management is as follows:

* Some alarms or abnormal operations, network performance metrics are reported from the network. Incident management agent receives these alarms/abnormal operations/metrics and analyzes the impact of these alarms on network services. If the analysis result indicates that network services are affected, an incident will be reported to the client.

* Incident management client receives the incident raised by agent, and acknowledge it. Client may invoke the 'incident diagnose' rpc to diagnose this incident to find the root causes.

* If the root causes have been found, the client can resolve this incident by invoking the 'incident resolve' rpc operation, dispatching a ticket or using other functions (e.g. routing calculation,configuration)

5. Functional Interface Requirements between the Client and the Agent

5.1. Incident Detection

In alarm-centric solution, although alarms are processed (based on manual rules or preconfigured rule) before being sent to the network OSS, multiple alarms are still sent to the network OSS. Whether these alarms have impact on network services and how much of the impact they created, it highly depends on the network OSS to analyze, which affects the efficiency of network maintenance.

```
           +---------------+
        +--|  Incident1    |
        |  +--+------------+
        |     |  +------------+
        |     +--+  alarm1    |
        |     |  +------------+
        |     |
        |     |  +------------+
        |     +--+  alarm2    |
        |     |  +------------+
        |     |
        |     |  +------------+
        |     +--+  alarm3    |
        |        +------------+
        |  +---------------+
        +--|  Incident2    |
        |  +--+------------+
        |     |  +------------+
        |     +--+  metric1   |
        |     |  +------------+
        |     |  +------------+
        |     +--+  metric2   |
        |        +------------+
        |
        |  +---------------+
        +--|  Incident3    |
           +--+------------+
              |  +------------+
              +--+ alarm1     |
              |  +------------+
              |
              |  +------------+
              +--| metric1    |
                 +------------+
```

Figure 2: Incident Detection

The incident management agent MUST be capable of detecting incidents. It can analyze the impact on network services from numerous alarms or monitor network service quality. Once the network service quality does not meet expectations, the incident agent MUST report the incident.

As described in Figure 2, multiple alarms, metrics, or hybrid can be aggregated into an incident after analysis. Each incident is associated with network services.

```
                        +----------------------+
                        |                      |
                        |     Orchestrator     |
                        |                      |
                        +----+-----------------+
                             ^VPN A Unavailable
                             |
                         +---+----------------+
                         |                    |
                         |     Controller     |
                         |                    |
                         |                    |
                         +-+-+-+-----+--+-----+
                         ^ ^            ^
                     IGP | |Interface   |IGP Peer
                    Down | |Down        | Abnormal
                         | |            |
        VPN A            | |            |
       +-----------------+-+------------+------------------*
       | \  +---+       ++-++         +-+-+        +---+  /|
       |  \ |   |       |   |         |   |        |   | / |
       |   \|PE1+-------| P1+X--------|P2 +--------|PE2|/  |
       |    +---+       +---+         +---+        +---+   |
       +---------------------------------------------------+
```

Figure 3: Example 1 of Incident Detection

As described in Figure 3, vpn a is deployed from PE1 to PE2, if a interface of P1 is going down, many alarms are triggered, such as interface down, igp down, and igp peer abnormal from P2. These alarms are aggregated and analyzed by controller, and the incident 'vpn unavailable' is triggered by the controller.

```
                       +----------------------+
                       |                      |
                       |     Orchestrator     |
                       |                      |
                       +----+-----------------+
                            ^VPN A Degradation
                            |
                        +---+----------------+
                        |                    |
                        |     controller     |
                        |                    |
                        |                    |
                        +-+-+-+-----+--+-----+
                          ^            ^
                          |Packet      |Path Delay
                          |Loss        |
                          |            |
       VPN A              |            |
      +-------------------+------------+-------------------+
      | \  +---+       ++-++         +-+-+        +---+  / |
      |  \ |   |       |   |         |   |        |   | /  |
      |   \|PE1+-------|P1 +---------|P2 +--------|PE2|/   |
      |    +---+       +---+         +---+        +---+    |
      +----------------------------------------------------+
```

Figure 4: Example 2 of Incident Detection

As described in Figure 4, controller collect the network metrics from network elements, it finds the packet loss of P1 and the path delay of P2 exceed the thresholds, an incident 'VPN A degradation' may be triggered after analysis.

5.2. Incident Diagnosis

After an incident is reported to the incident management client, the client MAY diagnose the incident to determine the root cause. Some diagnosis operations may affect the running network services. The client can choose not to perform that diagnosis operation after determining the impact is not trivial. The incident management agent can also perform self-diagnosis. However, the self-diagnosis MUST not affect the running network services. Possible diagnosis methods include link reachability detection, link quality detection, alarm/log analysis, and short-term fine-grained monitoring of network quality metrics, etc.

5.3. Incident Resolution

After the root cause is diagnosed, the client MAY resolve the incident. The client MAY choose resolve the incident by invoking other functions, such as routing calculation function, configuration function, dispatching a ticket or asking the agent to resolve it. Generally, the client would attempt to directly resolve the root cause. If the root cause cannot be resolved, an alternative solution SHOULD be required. For example, if an incident caused by a physical component failure, it cannot be automatically resolved, the standby link can be used to bypass the faulty component.

If the incident has been resolved, the client MAY indicate the agent to change the incident status to 'cleared'. If the incident is resolved by the agent, this indicator is unnecessary.

Incident resolution may affect the running network services. The client can choose not to perform those operations after determining the impact is not trivial.

6. Incident Data Model Concepts

6.1. Identifying the Incident Instance

An incident instance is associated with the specific network services instance and an incident name. An incident ID is used as an identifier of an incident instance, if an incident instance is detected, a new incident ID is created. The incident ID MUST be unique in the whole system.

6.2. The Incident Lifecycle

6.2.1. Incident Instance Lifecycle

From an incident instance perspective, an incident can have the following lifecycle: 'raised', 'updated', 'cleared'. When an incident is generated, the status is 'raised'. If the status changes after the incident is generated, (for example, self-diagnosis, diagnosis command issued by the client, or any other condition causes the status to change but does not reach the 'cleared' level.) , the status changes to 'updated'. When an incident is successfully resolved, the status changes to 'cleared'.

6.2.2. Operator Incident Lifecycle

From an operator perspective, the lifecycle of an incident instance includes ’acknowledged’, ’diagnosed’, and ’resolved’. When an incident instance is generated, the operator SHOULD acknowledge the incident. And then the operator attempts to diagnose the incident (for example, find out the root cause and affected components). Diagnosis is not mandatory. If the root cause and affected components are known when the incident is generated, diagnosis is not required. After locating the root cause and affected components, operator can try to resolve the incident.

7. Incident Data Model

7.1. Overview

```
      module: ietf-incident
        +--ro incidents
           +--ro incident* [incident-id]
              +--ro incident-id string
              +--ro csn uint64
              +--ro service-instance* string
              +--ro name string
              +--ro type enumeration
              +--ro domain identityref
              +--ro priority incident-priority
              +--ro status? enumeration
              +--ro ack-status? enumeration
              +--ro category identityref
              +--ro tenant? string
              +--ro detail? string
              +--ro resolve-suggestion? string
              +--ro sources
              |  ...
              +--ro root-causes
              |  ...
              +--ro events
              |  ...
              +--ro raise-time? yang:date-and-time
              +--ro occur-time? yang:date-and-time
              +--ro clear-time? yang:date-and-time
              +--ro ack-time? yang:date-and-time
              +--ro last-updated? yang:date-and-time
        rpcs:
          +---x incident-acknowledge
          |  ...
          +---x incident-diagnose
          |  ...
          +---x incident-resolve
             ...
        notifications:
          +---n incident-notification
             +--ro incident-id? string
             ...
```

7.2. Incident Notifications

```
     notifications:
         +---n incident-notification
            +--ro incident-id? string
            +--ro csn uint64
            +--ro service-instance* string
            +--ro name string
            +--ro type enumeration
            +--ro domain identityref
            +--ro priority incident-priority
            +--ro status? enumeration
            +--ro ack-status? enumeration
            +--ro category identityref
            +--ro tenant? string
            +--ro detail? string
            +--ro resolve-suggestion? string
            +--ro sources
            |  +--ro source* [node]
            |     +--ro node
            |             -> /nw:networks/nw:network/nw:node/nw-inv:name
            |     +--ro resource* [name]
            |        +--ro name al:resource
            +--ro root-causes
            |  +--ro root-cause* [node]
            |     +--ro node
            |             -> /nw:networks/nw:network/nw:node/nw-inv:name
            |     +--ro resource* [name]
            |     |  +--ro name al:resource
            |     |  +--ro cause-name? string
            |     |  +--ro detail? string
            |     +--ro cause-name? string
            |     +--ro detail? string
            +--ro events
            |  +--ro event* [type original-node]
            |     +--ro type enumeration
            |     +--ro original-node union
            |     +--ro is-root? boolean
            |     +--ro (event-type-info)?
            |        +--:(alarm)
            |        |  +--ro alarm
            |        |     +--ro resource? leafref
            |        |     +--ro alarm-type-id? leafref
            |        |     +--ro alarm-type-qualifier? leafref
            |        +--:(notification)
            |        +--:(log)
            |        +--:(KPI)
            |        +--:(unknown)
            +--ro time? yang:date-and-time
```

A general notification, incident-notification, is provided here. When an incident instance is detected, the notification will be sent. After a notification is generated, if the incident management agent performs self diagnosis or the client uses the interfaces provided by the incident management agent to deliver diagnosis and resolution actions, the notification update behavior is triggered, for example, the root cause objects and affected objects are updated. When an incident is successfully resolved, the status of the incident would be set to 'cleared'.

7.3. Incident Acknowledge

```
+---x incident-acknowledge
|  +---w input
|  |  +---w incident-id* string
```

After an incident is generated, updated, or cleared, (In some scenarios where automatic diagnosis and resolution are supported, the status of an incident may be updated multiple times or even automatically resolved.) The operator needs to confirm the incident to ensure that the client knows the incident.

The incident-acknowledge rpc can confirm multiple incidents at a time

7.4. Incident Diagnose

```
+---x incident-diagnose
|  +---w input
|  |  +---w incident-id* string
|  +--ro output
|     +--ro incident* [incident-id]
|        +--ro incident-id? string
|        +--ro (result)?
|           +--:(success)
|           |  +--ro service-instance? string
|           |  +--ro name? string
|           |  +--ro domain? identityref
|           |  +--ro priority? incident-priority
|           |  +--ro impact? enumeration
|           |  +--ro status? enumeration
|           |  +--ro ack-status? enumeration
|           |  +--ro category? identityref
|           |  +--ro tenant? string
|           |  +--ro detail? string
|           |  +--ro resolve-suggestion? string
|           |  +--ro sources
|           |  |  +--ro source* [node]
|           |  |     +--ro node? leafref
```

```
       |           |  |     +--ro resource* [name]
       |           |  |        +--ro name? al:resource
       |           |  +--ro root-causes
       |           |  |  +--ro root-cause* [node]
       |           |  |     +--ro node? leafref
       |           |  |     +--ro resource* [name]
       |           |  |     |  +--ro name? al:resource
       |           |  |     |  +--ro cause-name? string
       |           |  |     |  +--ro detail? string
       |           |  |     +--ro cause-name? string
       |           |  |     +--ro detail? string
       |           |  +--ro affects
       |           |  |  +--ro affect* [node]
       |           |  |     +--ro node? leafref
       |           |  |     +--ro resource* [name]
       |           |  |     |  +--ro name? al:resource
       |           |  |     |  +--ro state? enumeration
       |           |  |     |  +--ro detail? string
       |           |  |     +--ro state? enumeration
       |           |  |     +--ro detail? string
       |           |  +--ro links
       |           |  |  +--ro link* leafref
       |           |  +--ro events
       |           |  |  +--ro event* [type original-node]
       |           |  |     +--ro type? enumeration
       |           |  |     +--ro original-node? union
       |           |  |     +--ro is-root? boolean
       |           |  |     +--ro (event-type-info)?
       |           |  |        +--:(alarm)
       |           |  |        |  +--ro alarm
       |           |  |        |     +--ro resource? leafref
       |           |  |        |     +--ro alarm-type-id? leafref
       |           |  |        |     +--ro alarm-type-qualifier? leafref
       |           |  |        +--:(notification)
       |           |  |        +--:(log)
       |           |  |        +--:(KPI)
       |           |  |        +--:(unknown)
       |           |  +--ro time? yang:date-and-time
       |           +--:(failure)
       |              +--ro error-code? string
       |              +--ro error-message? string
```

After an incident is generated, incident diagnose rpc can be used to diagnose the incident and locate the root causes. Diagnosis can be performed on some detection tasks, such as BFD detection, flow detection, telemetry collection, short-term threshold alarm, configuration error check, or test packet injection.

If the diagnosis is successful, the latest status of the incident will be returned and a notification of the incident update will be triggered. If the diagnosis fails, error code and error message will be returned.

7.5. Incident Resolution

```
     +---x incident-resolve
        +---w input
        |  +---w incident* [incident-id]
        |     +---w incident-id
        |             -> /inc:incidents/inc:incident/inc:incident-id
        |     +---w resolved? empty
        +--ro output
           +--ro incident* [incident-id]
              +--ro incident-id string
              +--ro (result)?
                 +--:(success)
                 |  +--ro success? empty
                 |  +--ro time? yang:date-and-time
                 +--:(failure)
                    +--ro error-code? string
                    +--ro error-message? string
```

After the root cause and impact are determined, incident-resolve rpc can be used to resolve the incident (if the agent can resolve it) or indicate the incident instances have been resolved by other means. How to resolve an incident instance is out of the scope of this document.

Incident resolve rpc allows multiple incident instances to be resolved at a time. If an incident instance is successfully resolved, the success flag and resolve time will be returned, and a notification will be triggered to update the incident status to 'cleared'. If an incident fails to be resolved, an error code and an error message will be returned. If the incident content is changed during this process, a notification update will be triggered.

8. Incident Management YANG Module

```
<CODE BEGINS>
       file="ietf-incident@2023-03-13.yang"
   module ietf-incident {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-incident";
     prefix inc;
     import ietf-yang-types {
       prefix yang;
```

```
          reference
            "RFC 6991: Common YANG Data Types";
        }
        import ietf-network {
          prefix nw;
          reference
            "RFC 8345: A YANG Data Model for Network Topologies";
        }
        import ietf-network-inventory {
          prefix nw-inv;
          reference
            "draft-wzwb-opsawg-network-inventory-management-01:
            An Inventory Management Model for Enterprise Networks";
        }
        import ietf-alarms {
          prefix al;
          reference
            "RFC 8632: A YANG Data Model for Alarm Management";
        }
        organization
          "IETF OPSAWG Working Group";
        contact
          "WG Web:   &lt;https://datatracker.ietf.org/wg/opsawg/&gt;
           WG List:  &lt;mailto:opsawg@ietf.org&gt;
           Author:   Chong Feng  &lt;mailto:frank.fengchong@huawei.com&gt;
           Author:   Tong Hu  &lt;mailto:hutong@cmhi.chinamobile.com&gt;
           Author:   Luis Miguel Contreras Murillo &lt;mailto:
                     luismiguel.contrerasmurillo@telefonica.com&gt;;
           Author :   Qin Wu   &lt;mailto:bill.wu@huawei.com&gt;
           Author:   ChaoDe Yu   &lt;mailto:yuchaode@huawei.com&gt;";

       description
          "This module defines the interfaces for incident management
           lifecycle.

           This module is intended for the following use cases:
           * incident lifecycle management:
             - incident report: report incident instance to client
                                when an incident instance is detected.
             - incident acknowledge: acknowledge an incident instance.
             - incident diagnose: diagnose an incident instance.
             - incident resolve: resolve an incident instance.

           Copyright (c) 2022 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 Revised 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 2023-03-13 {
          description "initial version";
          reference "RFC XXX: Yang module for incident management.";
        }
        //identities
        identity incident-domain {
          description "The abstract identity to indicate the domain of
                       an incident.";
        }
        identity single-domain {
          base incident-domain;
          description "single domain.";
        }
        identity access {
          base single-domain;
          description "access domain.";
        }
        identity ran {
          base access;
          description "radio access network domain.";
        }
        identity transport {
          base single-domain;
          description "transport domain.";
        }
        identity otn {
          base transport;
          description "optical transport network domain.";
        }
        identity ip {
          base single-domain;
          description "ip domain.";
        }
        identity ptn {
          base ip;
          description "packet transport network domain.";
        }

        identity cross-domain {
          base incident-domain;
          description "cross domain.";
        }
        identity incident-category {
          description "The abstract identity for incident category.";
        }
        identity device {
          base incident-category;
          description "device category.";
        }
        identity power-enviorment {
          base device;
          description "power system category.";
        }
        identity device-hardware {
          base device;
          description "hardware of device category.";
        }
        identity device-software {
          base device;
          description "software of device category";
        }
        identity line {
          base device-hardware;
          description "line card category.";
        }
        identity maintenance {
          base incident-category;
          description "maintenance category.";
        }
        identity network {
          base incident-category;
          description "network category.";
        }
        identity protocol {
          base incident-category;
          description "protocol category.";
        }
        identity overlay {
          base incident-category;
          description "overlay category";
        }
        identity vm {
          base incident-category;
          description "vm category.";
        }

        //typedefs
        typedef incident-priority {
          type enumeration {
            enum critical {
              description "the incident MUST be handled immediately.";
            }
            enum high {
              description "the incident should be handled as soon as
                           possible.";
            }
            enum medium {
              description "network services are not affected, or the
                           services are slightly affected,but corrective
                           measures need to be taken.";
            }
            enum low {
              description "potential or imminent service-affecting
                           incidents are detected,but services are
                           not affected currently.";
            }
          }
          description "define the priority of incident.";
        }
        typedef node-ref {
          type leafref {
            path "/nw:networks/nw:network/nw:node/nw-inv:name";
          }
          description "reference a network node.";
        }
        //groupings
        grouping resources-info {
          description "the grouping which defines the network
                       resources of a node.";
          leaf node {
            type node-ref;
            description "reference to a network node.";
          }
          list resource {
            key name;
            description "the resources of a network node.";
            leaf name {
               type al:resource;
               description "network resource name.";
            }
          }
        }

        grouping incident-time-info {
          description "the grouping defines incident time information.";
          leaf raise-time {
            type yang:date-and-time;
            description "the time when an incident instance is raised.";
          }
          leaf occur-time {
            type yang:date-and-time;
            description "the time when an incident instance is occured.
                         It’s the occur time of the first event during
                         incident detection.";
          }
          leaf clear-time {
            type yang:date-and-time;
            description "the time when an incident instance is
                         resolved.";
          }
          leaf ack-time {
            type yang:date-and-time;
            description "the time when an incident instance is
                         acknowledged.";
          }
          leaf last-updated {
            type yang:date-and-time;
            description "the latest time when an incident instance is
                         updated";
          }
        }

        grouping incident-info {
          description "the grouping defines the information of an
                       incident.";
          leaf csn {
            type uint64;
            mandatory true;
            description "The sequence number of the incident instance.";
          }
          leaf-list service-instance {
            type string;
            description "the related network service instances of
                         the incident instance.";
          }
          leaf name {
            type string;
            mandatory true;
            description "the name of an incident.";
          }
          leaf type {
            type enumeration {
              enum fault {
                description "It indicates the type of the incident
                             is a fault, for example an interface
                             fails to work.";
              }
              enum potential-risk {
                description "It indicates the type of the incident
                             is a potential risk, for example high
                             CPU rate may cause a fault in the
                             future.";
              }
            }
            mandatory true;
            description "The type of an incident.";
          }
          leaf domain {
            type identityref {
              base incident-domain;
            }
            mandatory true;
            description "the domain of an incident.";
          }
          leaf priority {
            type incident-priority;
            mandatory true;
            description "the priority of an incident instance.";
          }

          leaf status {
            type enumeration {
              enum raised {
                description "an incident instance is raised.";
              }
              enum updated {
                description "the information of an incident instance
                             is updated.";
              }
              enum cleared {
                description "an incident is cleared.";
              }
            }
            default raised;
            description "The status of an incident instance.";
          }
          leaf ack-status {
            type enumeration {
              enum acknowledged;
              enum unacknowledged;
            }
            default unacknowledged;
            description "the acknowledge status of an incident.";
          }
          leaf category {
            type identityref {
              base incident-category;
            }
            mandatory true;
            description "The category of an incident.";
          }

          leaf tenant {
            type string;
            description "the identifier of related tenant.";
          }
          leaf detail {
            type string;
            description "detail information of this incident.";
          }
          leaf resolve-suggestion {
            type string;
            description "The suggestion to resolve this incident.";
          }
          container sources {
            description "The source components.";
            list source {
              key node;
              uses resources-info;
              min-elements 1;
              description "The source components of incident.";
            }
          }

          container root-causes{
            description "The root cause objects.";
            list root-cause {
              key node;
              description "the root causes of incident.";
              grouping root-cause-info {
                description "The information of root cause.";
                leaf cause-name {
                  type string;
                  description "the name of cause";
                }
                leaf detail {
                  type string;
                  description "the detail information of the cause.";
                }
              }
              uses resources-info {
                augment resource {
                  description "augment root cause information.";
                  //if root cause object is a resource of a node
                  uses root-cause-info;
                }
              }
              //if root cause object is a node
              uses root-cause-info;
            }
          }
          container events {
            description "related event.";
            list event {
              key "type original-node";
              description "related event.";
              leaf type {
                type enumeration {
                  enum alarm {
                    description "alarm type";
                  }
                  enum notification {
                    description "notification type";
                  }
                  enum log {
                    description "log type";
                  }
                  enum KPI {
                    description "KPI type";
                  }
                  enum unknown {
                    description "unknown type";
                  }
                }
                description "event type.";
              }
              leaf original-node {
                type union {
                  type node-ref;
                  type empty;//self
                }
                description "the original node where the event occurs.";
              }
              leaf is-root {
                type boolean;
                default false;
                description "whether this event is the cause of
                              incident.";
              }
              choice event-type-info {
                description "event type information.";
                case alarm {
                  when "type = 'alarm'";
                  container alarm {
                    description "alarm type event.";
                    leaf resource {
                      type leafref {
                        path "/al:alarms/al:alarm-list/al:alarm"
                            +"/al:resource";
                      }
                      description "network resource.";
                      reference "RFC 8632: A YANG Data Model for Alarm
                                 Management";
                    }
                    leaf alarm-type-id {
                      type leafref {
                        path "/al:alarms/al:alarm-list/al:alarm"
                            +"[al:resource = current()/../resource]"
                            +"/al:alarm-type-id";
                      }
                      description "alarm type id";
                      reference "RFC 8632: A YANG Data Model for Alarm
                                  Management";
                    }
                    leaf alarm-type-qualifier {
                      type leafref {
                        path "/al:alarms/al:alarm-list/al:alarm"
                            +"[al:resource = current()/../resource]"
                            +"[al:alarm-type-id = current()/.."
                            +"/alarm-type-id]/al:alarm-type-qualifier";
                      }
                      description "alarm type qualitifier";
                      reference "RFC 8632: A YANG Data Model for Alarm
                                 Management";
                    }
                  }
                }
                case notification {
                  //TODO
                }
                case log {
                //TODO
                }
                case KPI {
                //TODO
                }
                case unknown {
                //TODO
                }
              }
            }

          }

        }

        //data definitions
        container incidents {
          config false;
          description "the information of incidents.";
          list incident {
            key incident-id;
            description "the information of incident.";
            leaf incident-id {
              type string;
              description "the identifier of an incident instance.";
            }
            uses incident-info;
            uses incident-time-info;
          }
        }

        // notifications
        notification incident-notification {
          description "incident notification. It will be triggered when
                       the incident is raised, updated or cleared.";
          leaf incident-id {
            type string;
            description "the identifier of an incident instance.";
          }
          uses incident-info;
          leaf time {
            type yang:date-and-time;
            description "occur time of an incident instance.";
          }
        }
        // rpcs
        rpc incident-acknowledge {
          description "This rpc can be used to acknowledge the specified
                       incidents.";
          input {
            leaf-list incident-id {
              type string;
              description "the identifier of an incident instance.";
            }
          }
        }
        rpc incident-diagnose {
          description "This rpc can be used to diagnose the specified
                       incidents.";
          input {
            leaf-list incident-id {
              type string;
              description
                "the identifier of an incident instance.";
            }
          }
          output {
            list incident {
              key incident-id;
              description "The entry of returned incidents.";
              leaf incident-id {
                type string;
                description
                  "the identifier of an incident instance.";
              }
              choice result {
                description "result information.";
                case success {
                  uses incident-info;
                  leaf time {
                    type yang:date-and-time;
                    description
                      "The update time of an incident.";
                  }
                }
                case failure {
                  leaf error-code {
                    type string;
                    description "error code";
                  }
                  leaf error-message {
                    type string;
                    description "error message";
                  }
                }
              }
            }
          }
        }

        rpc incident-resolve {
          description "This rpc can be used to resolve the specified
                       incidents. It also can be used to set the
                       incident instances are resolved if these incident
                       instances are resolved by external system.";
          input {
            list incident {
              key incident-id;
              min-elements 1;
              description "incident instances.";
              leaf incident-id {
                type leafref {
                  path "/inc:incidents/inc:incident/inc:incident-id";
                }
                description
                  "the identifier of an incident instance.";
              }
              leaf resolved {
                type empty;
                description "indicate the incident instance has
                             been resolved.";
              }

            }
          }
          output {
            list incident {
              key incident-id;
              description "incident instances";
              leaf incident-id {
                type string;
                description "the identifier of incident instance";
              }
              choice result {
                description "result information";
                case success {
                  leaf success {
                    type empty;
                    description "reslove incident instance
                                 successfully";
                  }
                  leaf time {
                    type yang:date-and-time;
                    description "The resolved time of an incident.";
                  }
                }
                case failure {
                  leaf error-code {
                    type string;
                    description "error code";
                  }
                  leaf error-message {
                    type string;
                    description "error message.";
                  }
                }
              }
            }
          }
        }
      }
   <CODE ENDS>
```

9. IANA Considerations

9.1. The "IETF XML" Registry

This document registers one XML namespace URN in the 'IETF XML registry', following the format defined in [RFC3688].

```
   URI: urn:ietf:params:xml:ns:yang:ietf-incident
   Registrant Contact: The IESG.
   XML: N/A, the requested URIs are XML namespaces.
```

9.2. The "YANG Module Names" Registry

This document registers one module name in the 'YANG Module Names' registry, defined in [RFC6020].

```
   name: ietf-incident
   prefix: inc
   namespace: urn:ietf:params:xml:ns:yang:ietf-incident
   RFC: XXXX
   // RFC Ed.: replace XXXX and remove this comment
```

10. Security Considerations

The YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocol 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 Network Configuration Access Control Model (NACM) [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.

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:

Some of the readable data nodes in this YANG module 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. These are the subtrees and data nodes and their sensitivity/vulnerability:

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

11. Contributors

Aihua Guo
Futurewei Technologies
Email: aihuaguo.ietf@gmail.com

12. Acknowledgments

The authors would like to thank Mohamed Boucadair, Zhidong Yin, Guoxiang Liu, Haomian Zheng, YuanYao for their valuable comments and great input to this work.

13. References

13.1. 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>.

[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>.

[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>.

[RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <https://www.rfc-editor.org/info/rfc8345>.

[RFC8632] Vallin, S. and M. Bjorklund, "A YANG Data Model for Alarm Management", RFC 8632, DOI 10.17487/RFC8632, September 2019, <https://www.rfc-editor.org/info/rfc8632>.

13.2. Informative References

[I-D.ietf-opsawg-yang-vpn-service-pm]
Wu, B., Wu, Q., Boucadair, M., de Dios, O. G., and B. Wen, "A YANG Model for Network and VPN Service Performance Monitoring", Work in Progress, Internet-Draft, draft-ietf-opsawg-yang-vpn-service-pm-15, 11 November 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-yang-vpn-service-pm-15>.

[I-D.wzwb-opsawg-network-inventory-management]
Wu, B., Zhou, C., Wu, Q., and M. Boucadair, "An Inventory Management Model for Enterprise Networks", Work in Progress, Internet-Draft, draft-wzwb-opsawg-network-inventory-management-01, 10 February 2023, <https://datatracker.ietf.org/doc/html/draft-wzwb-opsawg-network-inventory-management-01>.

[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>.

[RFC8969] Wu, Q., Ed., Boucadair, M., Ed., Lopez, D., Xie, C., and L. Geng, "A Framework for Automating Service and Network Management with YANG", RFC 8969, DOI 10.17487/RFC8969, January 2021, <https://www.rfc-editor.org/info/rfc8969>.

[W3C-Trace-Context]
W3C, "W3C Recommendation on Trace Context", 23 November 2021, <https://www.w3.org/TR/2021/REC-trace-context-1-20211123/>.

Authors' Addresses

Chong Feng (editor)
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: frank.fengchong@huawei.com

Tong Hu
China Mobile (Hangzhou) Information Technology Co., Ltd
Building A01, 1600 Yuhangtang Road, Wuchang Street, Yuhang District
Hangzhou
ZheJiang, 311121
China
Email: hutong@cmhi.chinamobile.com

Luis Miguel Contreras Murillo
Telefonica I+D
Madrid
Spain
Email: luismiguel.contrerasmurillo@telefonica.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: bill.wu@huawei.com

Chaode Yu
Huawei
Email: yuchaode@huawei.com

netmod   J. Haas
Internet-Draft   Juniper Networks
Intended status: Standards Track   30 January 2023
Expires: 3 August 2023

Representing Unknown YANG bits in Operational State
draft-haas-netmod-unknown-bits-01

Abstract

Protocols frequently have fields where the contents are a series of bits that have specific meaning. When modeling operational state for such protocols in YANG, the ’bits’ YANG built-in type is a natural method for modeling such fields. The YANG ’bits’ built-in type is best suited when the meaning of a bit assignment is clear.

When bits that are currently RESERVED or otherwise unassigned by the protocol are received, being able to model them is necessary in YANG operational models. This cannot be done using the YANG ’bits’ built-in type without assigning them a name. However, YANG versioning rules do not permit renaming of named bits.

This draft proposes a methodology to represent unknown bits in YANG operational models and creates a YANG typedef to assist in uniformly naming such unknown bits.

Status of This Memo

Copyright Notice

Table of Contents

1. Requirements Language

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. Modeling Protocol Bit Vectors in YANG

Protocols frequently will have bit vectors as fields. Not all bits in such bit vectors are assigned during the specification of the protocol. These unassigned bits are typically made RESERVED and are used at a later date to provide for new features.

The YANG 'bits' built-in type (Section 9.7 of [RFC7950]) can be used to provide a "named bit" mapping to currently assigned bits in such fields. The representation format of 'bits' is "a space-separated list of the names of the bits that are set". However, when no assignment has been made for a bit position, nothing will be rendered.

There are operational needs for displaying received bits that may not be part of known assignments in the protocol. One such example is debugging behavior when unexpected bits have been sent in the protocol. This may occur when interacting with a version of the protocol that has assigned a previously unassigned bit.

One way to model such a scenario is to have one YANG leaf that covers known bit assignments, and have a subsequent YANG leaf contain unknown bits.

3. Modeling Unknown Bits

3.1. Example of Issue: Modeling BGP's Graceful Restart Flags

BGP's Graceful Restart Capability (Section 3 of [RFC4724]) contains a Restart Flags field that is four bits wide. Its definition is copied below:

```
 0 1 2 3
+-+-+-+-+
|R|Resv.|
+-+-+-+-+
```

Figure 1: BGP Graceful Restart Flags

The 'R' (Restart State) bit has been assigned in RFC 4724. One way to model this (taken from [I-D.ietf-idr-bgp-model]) is:

```
     typedef graceful-restart-flags {
       type bits {
         bit restart {
           position 0;
           description
             "The most significant bit is defined as the Restart
              State (R) bit, [...]";
           reference
             "RFC 4724: Graceful Restart Mechanism for BGP,
              Section 3.";
         }
       }
       [...]
     }

     [...]

     leaf flags {
       type bt:graceful-restart-flags;
       description
         "Restart Flags advertised by the Graceful Restart
          Capability";
       reference
         "RFC 4724: Graceful Restart Mechanism for BGP, Section 3.";
     }
```

Figure 2: BGP Graceful Restart Flags

[RFC8538] later assigns bit position 1 to the 'N' flag, updating the set of flags used in this field:

```
                                  0 1 2 3
                                 +-+-+-+-+
                                 |R|N|   |
                                 +-+-+-+-+
```

Figure 3: BGP Graceful Restart Flags, Revised by RFC 8538

YANG module versioning rules would require the graceful-restart-flags typedef to be updated. For protocol well-known fields, this encourages such typedefs to be IANA-maintained for ease of update. A revised typedef may resemble:

```
     typedef graceful-restart-flags {
       type bits {
         bit restart {
           position 0;
           description
             "The most significant bit is defined as the Restart
              State (R) bit, [...]";
           reference
             "RFC 4724: Graceful Restart Mechanism for BGP,
              Section 3.";
         }
         bit notification {
           position 1;
           description
             "The second most significant bit is defined in [RFC 8538]
              as the Graceful Notification ('N') bit. [...]";
           reference
             "RFC 8538: Notification Message Support for BGP Graceful
              Restart, Section 2.";
         }
       }
     }
```

Figure 4: Revised BGP Graceful Restart Flags Typedef

Consider a router supporting the old typedef receiving a BGP Graceful Restart Capability containing both the 'R' and 'N' bits in the BGP protocol. In that typedef, the "flags" leaf could only represent position 0, the "restart" named bit. The implementation couldn't represent that the 'N' bit was sent in the protocol.

```
<flags>restart</flags>
```

Figure 5: Flags for 'R' and 'N' bits with original leaves and typedef

3.2. Defining Unknown Bits

One solution to modeling unknown bits is to have a subsequent leaf whose purposes is only to model unknown bit mappings. When the protocol does not send the unassigned bits, this leaf would be absent in the output of the operational state.

Using the example where only the 'R' bit was defined, one way to model this would be:

```
        typedef  unknown-flags {
          type bits {
            bit unknown-1 {
              position 1;
              description
                "Bit 1 was received but is currently RESERVED.";
            }
            bit unknown-2 {
              position 2;
              description
                "Bit 2 was received but is currently RESERVED.";
            }
            bit unknown-3 {
              position 3;
              description
                "Bit 3 was received but is currently RESERVED.";
            }
          }
          description
            "When a bit is exchanged in the Graceful Restart Flags
             field that is unknown to this module, their bit position
             is rendered using the associated unknown bit.";
          reference
            "RFC 4724: Graceful Restart Mechanism for BGP, Section 3.";
        }
        leaf unknown-flags {
          type unknown-flags;
          description
            "Restart Flags advertised by the Graceful Restart
             Capability";
          reference
            "RFC 4724: Graceful Restart Mechanism for BGP, Section 3.";
        }
```

Figure 6: BGP Graceful Restart Specific Unknown Bits

If the router using the above modeling received a BGP Graceful Restart Capability containing both the ’R’ and the ’N’ bits, it would now be rendered:

```
            <flags>restart</flags>
            <unknown-flags>unknown-1</unknown-flags>
```

Figure 7: Flags for ’R’ and ’N’ bits with new leaves and typedefs

Deleting bit assignments in later versions of the model is not permitted by current YANG versioning rules. The only purpose of such unknown named bits is to represent fields that may later be assigned during maintenance of the protocol.

For example, when position 1, "bit notification" is assigned, the same example scenario would then render as:

```
            <flags>restart unknown</flags>
```

Figure 8: Flags for 'R' and 'N' bits with new leaves and updated typedef

3.3. Consistently Modeling Unknown Bits

Each YANG module requiring this pattern to represent unknown bits could define its own protocol-specific typedefs for the appropriate number of unknown bits for their fields. However, there is operational benefit to use a consistent pattern for such unknown bits. A common typedef for this purpose, "unknown-bits", is defined in the next section.

The unknown-bits typedef defines 64 bits of unknown bits. Considering the example for the BGP Graceful Restart Flags bits where only 4 bits are present in the field, 64 bits for the typedef are not a problem. Only the bits received in the protocol that aren't recognized would be represented in the protocol-specific "unknown-flags" leaf, or similar.

Here's an example usage of this typedef using the prior "unknown-flags" leaf:

```
     include ietf-yang-unknown-bit-types {
       prefix yang-ubt;
     }
     leaf unknown-flags {
       type ubt:unknown-bits;
       description
         "When a bit is exchanged in the Graceful Restart Flags
          field that is unknown to this module, their bit position
          is rendered using the associated unknown bit.";
       reference
         "RFC 4724: Graceful Restart Mechanism for BGP, Section 3.";
     }
```

Figure 9: BGP Graceful Restart Specific Unknown Bits with Typedef

4. IETF YANG Unknown Bit Types Module

```
  module ietf-yang-unknown-bit-types {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-yang-unknown-bit-types";
    prefix yang-ubt;

    // meta

    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:   Jeffrey Haas
                 <mailto:jhaas@juniper.net>";

    description
      "This module contains data definitions for modeling operational
       state that would normally be represented using the YANG 'bits'
       type, but currently no known mapping for that bit position is
       registered.

       Copyright (c) 2023 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 2023-01-25 {
      description
        "Initial Version";
      reference
        "RFC XXXX: YANG module for unknown bit types.";
    }

    /*
     * Typedefs
     */

    typedef unknown-bits {
      type bits {
        bit unknown-0 {
          position 0;
          description
            "Bit 0 is unknown.";
        }
        bit unknown-1 {
          position 1;
          description
            "Bit 1 is unknown.";
        }
        bit unknown-2 {
          position 2;
          description
            "Bit 2 is unknown.";
        }
        bit unknown-3 {
          position 3;
          description
            "Bit 3 is unknown.";
        }
        bit unknown-4 {
          position 4;
          description
            "Bit 4 is unknown.";
        }
        bit unknown-5 {
          position 5;
          description
            "Bit 5 is unknown.";
        }
        bit unknown-6 {
          position 6;
          description
            "Bit 6 is unknown.";
        }
        bit unknown-7 {
          position 7;
          description
```

```
            "Bit 7 is unknown.";
        }
        bit unknown-8 {
          position 8;
          description
            "Bit 8 is unknown.";
        }
        bit unknown-9 {
          position 9;
          description
            "Bit 9 is unknown.";
        }
        bit unknown-10 {
          position 10;
          description
            "Bit 10 is unknown.";
        }
        bit unknown-11 {
          position 11;
          description
            "Bit 11 is unknown.";
        }
        bit unknown-12 {
          position 12;
          description
            "Bit 12 is unknown.";
        }
        bit unknown-13 {
          position 13;
          description
            "Bit 13 is unknown.";
        }
        bit unknown-14 {
          position 14;
          description
            "Bit 14 is unknown.";
        }
        bit unknown-15 {
          position 15;
          description
            "Bit 15 is unknown.";
        }
        bit unknown-16 {
          position 16;
          description
            "Bit 16 is unknown.";
        }
        bit unknown-17 {
          position 17;
          description
            "Bit 17 is unknown.";
        }
        bit unknown-18 {
          position 18;
          description
            "Bit 18 is unknown.";
        }
        bit unknown-19 {
          position 19;
          description
            "Bit 19 is unknown.";
        }
        bit unknown-20 {
          position 20;
          description
            "Bit 20 is unknown.";
        }
        bit unknown-21 {
          position 21;
          description
            "Bit 21 is unknown.";
        }
        bit unknown-22 {
          position 22;
          description
            "Bit 22 is unknown.";
        }
        bit unknown-23 {
          position 23;
          description
            "Bit 23 is unknown.";
        }
        bit unknown-24 {
          position 24;
          description
            "Bit 24 is unknown.";
        }
        bit unknown-25 {
          position 25;
          description
            "Bit 25 is unknown.";
        }
        bit unknown-26 {
          position 26;
          description
            "Bit 26 is unknown.";
```

```
        }
        bit unknown-27 {
          position 27;
          description
            "Bit 27 is unknown.";
        }
        bit unknown-28 {
          position 28;
          description
            "Bit 28 is unknown.";
        }
        bit unknown-29 {
          position 29;
          description
            "Bit 29 is unknown.";
        }
        bit unknown-30 {
          position 30;
          description
            "Bit 30 is unknown.";
        }
        bit unknown-31 {
          position 31;
          description
            "Bit 31 is unknown.";
        }
        bit unknown-32 {
          position 32;
          description
            "Bit 32 is unknown.";
        }
        bit unknown-33 {
          position 33;
          description
            "Bit 33 is unknown.";
        }
        bit unknown-34 {
          position 34;
          description
            "Bit 34 is unknown.";
        }
        bit unknown-35 {
          position 35;
          description
            "Bit 35 is unknown.";
        }
        bit unknown-36 {
          position 36;
          description
            "Bit 36 is unknown.";
        }
        bit unknown-37 {
          position 37;
          description
            "Bit 37 is unknown.";
        }
        bit unknown-38 {
          position 38;
          description
            "Bit 38 is unknown.";
        }
        bit unknown-39 {
          position 39;
          description
            "Bit 39 is unknown.";
        }
        bit unknown-40 {
          position 40;
          description
            "Bit 40 is unknown.";
        }
        bit unknown-41 {
          position 41;
          description
            "Bit 41 is unknown.";
        }
        bit unknown-42 {
          position 42;
          description
            "Bit 42 is unknown.";
        }
        bit unknown-43 {
          position 43;
          description
            "Bit 43 is unknown.";
        }
        bit unknown-44 {
          position 44;
          description
            "Bit 44 is unknown.";
        }
        bit unknown-45 {
          position 45;
          description
            "Bit 45 is unknown.";
        }
        bit unknown-46 {
          position 46;
          description
            "Bit 46 is unknown.";
        }
        bit unknown-47 {
          position 47;
          description
            "Bit 47 is unknown.";
        }
        bit unknown-48 {
          position 48;
          description
            "Bit 48 is unknown.";
        }
        bit unknown-49 {
          position 49;
          description
            "Bit 49 is unknown.";
        }
        bit unknown-50 {
          position 50;
          description
            "Bit 50 is unknown.";
        }
        bit unknown-51 {
          position 51;
          description
            "Bit 51 is unknown.";
        }
        bit unknown-52 {
          position 52;
          description
            "Bit 52 is unknown.";
        }
        bit unknown-53 {
          position 53;
          description
            "Bit 53 is unknown.";
        }
        bit unknown-54 {
          position 54;
          description
            "Bit 54 is unknown.";
        }
        bit unknown-55 {
          position 55;
          description
```

```
              "Bit 55 is unknown.";
          }
          bit unknown-56 {
            position 56;
            description
              "Bit 56 is unknown.";
          }
          bit unknown-57 {
            position 57;
            description
              "Bit 57 is unknown.";
          }
          bit unknown-58 {
            position 58;
            description
              "Bit 58 is unknown.";
          }
          bit unknown-59 {
            position 59;
            description
              "Bit 59 is unknown.";
          }
          bit unknown-60 {
            position 60;
            description
              "Bit 60 is unknown.";
          }
          bit unknown-61 {
            position 61;
            description
              "Bit 61 is unknown.";
          }
          bit unknown-62 {
            position 62;
            description
              "Bit 62 is unknown.";
          }
          bit unknown-63 {
            position 63;
            description
              "Bit 63 is unknown.";
          }
        }
        description
          "Typedef describing 64 bits worth of unknown bits.  This can be
           used to model operational state that would normally be modeled
           using the YANG 'bits' type, but no registered bit has been
           created.";
```

```
    }
  }
```

Figure 10

5. IANA Considerations

This document registers one URI and one YANG module.

5.1. URI Registration

Following the format in the IETF XML registry [RFC3688] [RFC3688], the following registration is requested to be made:

```
URI: urn:ietf:params:xml:ns:yang:ietf-yang-unknown-bit-types
```

Figure 11

Registrant Contact: The IESG. XML: N/A, the requested URI is an XML namespace.

5.2. YANG Module Name Registration

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

```
name: ietf-yang-unknown-bit-types
namespace: urn:ietf:params:xml:ns:yang:ietf-yang-unknown-bit-types
prefix: yang-ubt
reference: RFC XXXX
```

Figure 12

6. Security Considerations

Lack of operational visibility for protocol state can make troubleshooting protocol issues more difficult. The mechanism defined in this document may help reduce the scope of such issues and potentially remove the security considerations such lack of operational visibility may cause.

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>.

[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>.

[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>.

7.2. 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>.

[RFC4724] Sangli, S., Chen, E., Fernando, R., Scudder, J., and Y. Rekhter, "Graceful Restart Mechanism for BGP", RFC 4724, DOI 10.17487/RFC4724, January 2007, <https://www.rfc-editor.org/info/rfc4724>.

[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>.

[RFC8538] Patel, K., Fernando, R., Scudder, J., and J. Haas, "Notification Message Support for BGP Graceful Restart", RFC 8538, DOI 10.17487/RFC8538, March 2019, <https://www.rfc-editor.org/info/rfc8538>.

[I-D.ietf-idr-bgp-model] Jethanandani, M., Patel, K., Hares, S., and J. Haas, "BGP YANG Model for Service Provider Networks", Work in Progress, Internet-Draft, draft-ietf-idr-bgp-model-15, 13 October 2022, <https://www.ietf.org/archive/id/draft-ietf-idr-bgp-model-15.txt>.

Acknowledgements

Martin Bjorklund provided a review on an early version of this document.

Thanks to Jurgen Schonwalder and the IETF netmod Working Group for their feedback.

Author's Address

Jeffrey Haas
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
United States of America
Email: jhaas@pfrc.org

ANIMA Working Group
Internet-Draft
Intended status: Standards Track
Expires: 11 August 2023

K. Watsen
Watsen Networks
M. Richardson
Sandelman Software
M. Pritikin
Cisco Systems
T. Eckert
Q. Ma
Huawei

7 February 2023

A Voucher Artifact for Bootstrapping Protocols
draft-ietf-anima-rfc8366bis-07

Abstract

This document defines a strategy to securely assign a pledge to an owner using an artifact signed, directly or indirectly, by the pledge’s manufacturer. This artifact is known as a "voucher".

This document defines an artifact format as a YANG-defined JSON or CBOR document that has been signed using a variety of cryptographic systems.

The voucher artifact is normally generated by the pledge’s manufacturer (i.e., the Manufacturer Authorized Signing Authority (MASA)).

This document updates RFC8366, merging a number of extensions into the YANG. The RFC8995 voucher request is also merged into this document.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-anima-rfc8366bis/.

Discussion of this document takes place on the anima Working Group mailing list (mailto:anima@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/anima/. Subscribe at https://www.ietf.org/mailman/listinfo/anima/.

Source for this draft and an issue tracker can be found at https://github.com/anima-wg/voucher.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines a strategy to securely assign a candidate device (pledge) to an owner using an artifact signed, directly or indirectly, by the pledge’s manufacturer, i.e., the Manufacturer Authorized Signing Authority (MASA). This artifact is known as the "voucher".

The voucher artifact is a JSON [RFC8259] document that conforms with a data model described by YANG [RFC7950]. It may also be serialized to CBOR [CBOR]. It is encoded using the rules defined in [RFC8259], and is signed using (by default) a CMS structure [RFC5652].

The primary purpose of a voucher is to securely convey a certificate, the "pinned-domain-cert" (and constrained variations), that a pledge can use to authenticate subsequent interactions. A voucher may be useful in several contexts, but the driving motivation herein is to support secure onboarding mechanisms. Assigning ownership is important to device onboarding mechanisms so that the pledge can authenticate the network that is trying to take control of it.

The lifetimes of vouchers may vary. In some onboarding protocols, the vouchers may include a nonce restricting them to a single use, whereas the vouchers in other onboarding protocols may have an indicated lifetime. In order to support long lifetimes, this document recommends using short lifetimes with programmatic renewal, see Section 8.1.

This document only defines the voucher artifact, leaving it to other documents to describe specialized protocols for accessing it. Some onboarding protocols using the voucher artifact defined in this document include: [ZERO-TOUCH], [SECUREJOIN], and [BRSKI].

2. Terminology

This document uses the following terms:

Artifact: Used throughout to represent the voucher as instantiated in the form of a signed structure.

Bootstrapping: See Onboarding.

Domain: The set of entities or infrastructure under common administrative control. The goal of the onboarding protocol is to enable a pledge to discover and join a domain.

Imprint: The process where a device obtains the cryptographic key material to identify and trust future interactions with a network. This term is taken from Konrad Lorenz's work in biology with new ducklings: "during a critical period, the duckling would assume that anything that looks like a mother duck is in fact their mother" [Stajano99theresurrecting]. An equivalent for a device is to obtain the fingerprint of the network's root certification authority certificate. A device that imprints on an attacker suffers a similar fate to a duckling that imprints on a hungry wolf. Imprinting is a term from psychology and ethology, as described in [imprinting].

Join Registrar (and Coordinator): A representative of the domain that is configured, perhaps autonomically, to decide whether a new device is allowed to join the domain. The administrator of the domain interfaces with a join registrar (and Coordinator) to control this process. Typically, a join registrar is "inside" its domain. For simplicity, this document often refers to this as just "registrar".

MASA (Manufacturer Authorized Signing Authority): The entity that, for the purpose of this document, signs the vouchers for a manufacturer's pledges. In some onboarding protocols, the MASA may have an Internet presence and be integral to the onboarding process, whereas in other protocols the MASA may be an offline service that has no active role in the onboarding process.

Onboarding: In previous documents the term "bootstrapping" has been used to describe mechanisms such as [BRSKI]. The industry has however, converged upon the term "onboarding", and this document uses that term throughout.

Owner: The entity that controls the private key of the "pinned-domain-cert" certificate conveyed by the voucher.

Pledge: The prospective device attempting to find and securely join a domain. When shipped, it only trusts authorized representatives of the manufacturer.

Registrar: See join registrar.

TOFU (Trust on First Use): Where a pledge device makes no security decisions but rather simply trusts the first domain entity it is contacted by. Used similarly to [RFC7435]. This is also known as the "resurrecting duckling" model.

Voucher: A signed statement from the MASA service that indicates to a pledge the cryptographic identity of the domain it should trust.

3. Requirements Language

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.

4. Survey of Voucher Types

A voucher is a cryptographically protected statement to the pledge device authorizing a zero-touch "imprint" on the join registrar of the domain. The specific information a voucher provides is influenced by the onboarding use case.

The voucher can impart the following information to the join registrar and pledge:

Assertion Basis: Indicates the method that protects the imprint (this is distinct from the voucher signature that protects the voucher itself). This might include manufacturer-asserted ownership verification, assured logging operations, or reliance on pledge endpoint behavior such as secure root of trust of measurement. The join registrar might use this information. Only some methods are normatively defined in this document. Other methods are left for future work.

Authentication of Join Registrar: Indicates how the pledge can authenticate the join registrar. This document defines a mechanism to pin the domain certificate. Pinning a symmetric key, a raw key, or "CN-ID" or "DNS-ID" information (as defined in [RFC6125]) is left for future work.

Anti-Replay Protections: Time- or nonce-based information to constrain the voucher to time periods or bootstrap attempts.

A number of onboarding scenarios can be met using differing combinations of this information. All scenarios address the primary threat of a Man-in-The-Middle (MiTM) registrar gaining control over the pledge device. The following combinations are "types" of vouchers:

	Assertion		Registrar ID		Validity	
Voucher Type	Logged	Verified	Trust Anchor	CN-ID or DNS-ID	RTC	Nonce
Audit	X		X			X
Nonceless Audit	X		X		X	
Owner Audit	X	X	X		X	X
Owner ID		X	X	X	X	
Bearer out-of-scope	X		wildcard	wildcard	optional	opt

Table 1

NOTE: All voucher types include a 'pledge ID serial-number' (not shown here for space reasons).

Audit Voucher: An Audit Voucher is named after the logging assertion mechanisms that the registrar then "audits" to enforce local policy. The registrar mitigates a MiTM registrar by auditing that an unknown MiTM registrar does not appear in the log entries. This does not directly prevent the MiTM but provides a response mechanism that ensures the MiTM is unsuccessful. The advantage is that actual ownership knowledge is not required on the MASA service.

Nonceless Audit Voucher: An Audit Voucher without a validity period statement. Fundamentally, it is the same as an Audit Voucher except that it can be issued in advance to support network partitions or to provide a permanent voucher for remote deployments.

Ownership Audit Voucher: An Audit Voucher where the MASA service has verified the registrar as the authorized owner. The MASA service mitigates a MiTM registrar by refusing to generate Audit Vouchers for unauthorized registrars. The registrar uses audit techniques to supplement the MASA. This provides an ideal sharing of policy decisions and enforcement between the vendor and the owner.

Ownership ID Voucher: Named after inclusion of the pledge's CN-ID or DNS-ID within the voucher. The MASA service mitigates a MiTM registrar by identifying the specific registrar (via WebPKI) authorized to own the pledge.

Bearer Voucher: A Bearer Voucher is named after the inclusion of a registrar ID wildcard. Because the registrar identity is not indicated, this voucher type must be treated as a secret and protected from exposure as any 'bearer' of the voucher can claim the pledge device. Publishing a nonceless bearer voucher effectively turns the specified pledge into a "TOFU" device with minimal mitigation against MiTM registrars. Bearer vouchers are out of scope.

5. Changes since RFC8366

[RFC8366] was published in 2018 during the development of [BRSKI], [ZERO-TOUCH] and other work-in-progress efforts. Since then the industry has matured significantly, and the in-the-field activity which this document supports has become known as _onboarding_ rather than _bootstrapping_.

The focus of [BRSKI] was onboarding of ISP and Enterprise owned wired routing and switching equipment, with IoT devices being a less important aspect. [ZERO-TOUCH] has focused upon onboarding of CPE equipment like cable modems and other larger IoT devices, again with smaller IoT devices being of less import.

Since [BRSKI] was published there is now a mature effort to do application-level onboarding of constrained IoT devices defined by The Thread and Fairhair (now OCF) consortia. The [cBRSKI] document has defined a version of [BRSKI] that is useable over constrained 802.15.4 networks using CoAP and DTLS, while [I-D.selander-ace-ake-authz] provides for using CoAP and EDHOC on even more constrained devices with very constrained networks.

[PRM] has created a new methodology for onboarding that does not depend upon a synchronous connection between the Pledge and the Registrar. This mechanism uses a mobile Registrar Agent that works to collect and transfer signed artifacts via physical travel from one network to another.

Both [cBRSKI] and [PRM] require extensions to the Voucher Request and the resulting Voucher. The new attribtes are required to carry the additional attributes and describe the extended semantics. In addition [cBRSKI] uses the serialization mechanism described in [YANGCBOR] to produce significantly more compact artifacts.

When the process to define [cBRSKI] and [PRM] was started, there was a belief that the appropriate process was to use the [RFC8040] _augment_ mechanism to further extend both the voucher request [BRSKI] and voucher [RFC8366] artifacts. However, [PRM] needs to extend an enumerated type with additional values and _augment_ can not do this, so that was initially the impetus for this document.

An attempt was then made to determine what would happen if one wanted to have a constrained version of the [PRM] voucher artifact. The result was invalid YANG, with multiple definitions of the core attributes from the [RFC8366] voucher artifact. After some discussion, it was determined that the _augment_ mechanism did not work, nor did it work better when [RFC8040] yang-data was replaced with the [RFC8971] structure mechanisms.

After significant discussion the decision was made to simply roll all of the needed extensions up into this document as "RFC8366bis".

This document therefore represents a merge of YANG definitions from [RFC8366], the voucher-request from [BRSKI], and then extensions to each of these from [cBRSKI], [CLOUD] and [PRM]. There are some difficulties with this approach: this document does not attempt to establish rigorous semantic definitions for how some attributes are to be used, referring normatively instead to the other relevant documents.

6. Voucher Artifact

The voucher’s primary purpose is to securely assign a pledge to an owner. The voucher informs the pledge which entity it should consider to be its owner.

This document defines a voucher that is a JSON-encoded or CBOR-encoded instance of the YANG module defined in Section 6.3 that has been, by default, CMS signed. [cBRSKI] definies how to encode with CBOR and sign the voucher with [COSE], while [jBRSKI] explains how to use [JWS] to do JSON signatures.

This format is described here as a practical basis for some uses (such as in NETCONF), but more to clearly indicate what vouchers look like in practice. This description also serves to validate the YANG data model.

[RFC8366] defined a media type and a filename extension for the CMS-encoded JSON type. Which type of voucher is expected is signaled (where possible) in the form of a MIME Content-Type, an HTTP Accept: header, or more mundane methods like use of a filename extension when a voucher is transferred on a USB key.

6.1. Tree Diagram

The following tree diagram illustrates a high-level view of a voucher document. The notation used in this diagram is described in [RFC8340]. Each node in the diagram is fully described by the YANG module in Section 6.3. Please review the YANG module for a detailed description of the voucher format.

```
module: ietf-voucher

  structure voucher:
    +-- voucher
       +-- created-on?                      yang:date-and-time
       +-- expires-on?                      yang:date-and-time
       +-- assertion?                       enumeration
       +-- serial-number                    string
       +-- idevid-issuer?                   binary
       +-- pinned-domain-cert?              binary
       +-- domain-cert-revocation-checks?   boolean
       +-- nonce?                           binary
       +-- pinned-domain-pubk?              binary
       +-- pinned-domain-pubk-sha256?       binary
       +-- last-renewal-date?               yang:date-and-time
       +-- est-domain?                      ietf:uri
       +-- additional-configuration?        ietf:uri
```

6.2. Examples

This section provides voucher examples for illustration purposes. These examples conform to the encoding rules defined in [RFC8259].

The following example illustrates an ephemeral voucher (uses a nonce). The MASA generated this voucher using the 'logged' assertion type, knowing that it would be suitable for the pledge making the request.

```
{
  "ietf-voucher:voucher": {
    "created-on": "2016-10-07T19:31:42Z",
    "assertion": "logged",
    "serial-number": "JADA123456789",
    "idevid-issuer": "base64encodedvalue==",
    "pinned-domain-cert": "base64encodedvalue==",
    "nonce": "base64encodedvalue=="
  }
}
```

The following example illustrates a non-ephemeral voucher (no nonce). While the voucher itself expires after two weeks, it presumably can be renewed for up to a year. The MASA generated this voucher using the 'verified' assertion type, which should satisfy all pledges.

```
{
  "ietf-voucher:voucher": {
    "created-on": "2016-10-07T19:31:42Z",
    "expires-on": "2016-10-21T19:31:42Z",
    "assertion": "verified",
    "serial-number": "JADA123456789",
    "idevid-issuer": "base64encodedvalue==",
    "pinned-domain-cert": "base64encodedvalue==",
    "domain-cert-revocation-checks": "true",
    "last-renewal-date": "2017-10-07T19:31:42Z"
  }
}
```

6.3. YANG Module

```
<CODE BEGINS>
module ietf-voucher {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-voucher";
  prefix vch;

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

  organization
    "IETF ANIMA Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/anima/>
     WG List:  <mailto:anima@ietf.org>
     Author:   Kent Watsen
               <mailto:kwatsen@juniper.net>
     Author:   Max Pritikin
               <mailto:pritikin@cisco.com>
     Author:   Michael Richardson
               <mailto:mcr+ietf@sandelman.ca>
     Author:   Toerless Eckert
               <mailto:tte+ietf@cs.fau.de>";
  description
    "This module defines the format for a voucher, which is
     produced by a pledge's manufacturer or delegate (MASA)
     to securely assign a pledge to an 'owner', so that the
     pledge may establish a secure connection to the owner's
     network infrastructure.

     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) 2023 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 8366; see the
     RFC itself for full legal notices.";

  revision 2023-01-10 {
    description
      "updated to support new assertion enumerated type";
    reference
      "RFC ZZZZ Voucher Profile for Bootstrapping Protocols";
  }

  // Top-level statement
  sx:structure voucher {
    uses voucher-artifact-grouping;
  }

  // Grouping defined for future augmentations

  grouping voucher-artifact-grouping {
    description
      "Grouping to allow reuse/extensions in future work.";
    container voucher {
      description
        "A voucher assigns a pledge to an owner using
         the (pinned-domain-cert) value.";
      leaf created-on {
        type yang:date-and-time;
        mandatory false;
        description
          "A value indicating the date this voucher was created.
           This node is primarily for human consumption and auditing.
           Future work MAY create verification requirements based on
           this node.";
      }
      leaf expires-on {
        type yang:date-and-time;
        must 'not(../nonce)';
        description
          "A value indicating when this voucher expires.  The node is
           optional as not all pledges support expirations, such as
           pledges lacking a reliable clock.

           If this field exists, then the pledges MUST ensure that
           the expires-on time has not yet passed. A pledge without
           an accurate clock cannot meet this requirement.

           The expires-on value MUST NOT exceed the expiration date
           of any of the listed 'pinned-domain-cert' certificates.";
      }
      leaf assertion {
        type enumeration {
          enum verified {
            value 0;
            description
              "Indicates that the ownership has been positively
               verified by the MASA (e.g., through sales channel
               integration).";
          }
          enum logged {
            value 1;
            description
              "Indicates that the voucher has been issued after
               minimal verification of ownership or control.  The
               issuance has been logged for detection of
               potential security issues (e.g., recipients of
               vouchers might verify for themselves that unexpected
               vouchers are not in the log).  This is similar to
               unsecured trust-on-first-use principles but with the
               logging providing a basis for detecting unexpected
               events.";
          }
          enum proximity {
            value 2;
            description
              "Indicates that the voucher has been issued after
               the MASA verified a proximity proof provided by the
               device and target domain.  The issuance has been
               logged for detection of potential security issues.
               This is stronger than just logging, because it
               requires some verification that the pledge and owner
               are in communication but is still dependent on
               analysis of the logs to detect unexpected events.";
          }
          enum agent-proximity {
            value 3;
            description
              "Indicates that the voucher has been issued
               after the MASA has verified a statement that
               a registrar agent has made contact with the device.
               This type of voucher is weaker than straight
               proximity, but stronger than logged.";
          }
        }
      }
      leaf serial-number {
        type string;
        mandatory true;
        description
          "The serial-number of the hardware.  When processing a
           voucher, a pledge MUST ensure that its serial-number
           matches this value.  If no match occurs, then the
```

```
              pledge MUST NOT process this voucher.";
         }
         leaf idevid-issuer {
           type binary;
           description
             "The Authority Key Identifier OCTET STRING (as defined in
              Section 4.2.1.1 of RFC 5280) from the pledge's IDevID
              certificate.  Optional since some serial-numbers are
              already unique within the scope of a MASA.
              Inclusion of the statistically unique key identifier
              ensures statistically unique identification of the
              hardware.
              When processing a voucher, a pledge MUST ensure that its
              IDevID Authority Key Identifier matches this value.  If no
              match occurs, then the pledge MUST NOT process this
              voucher.
              When issuing a voucher, the MASA MUST ensure that this
              field is populated for serial-numbers that are not
              otherwise unique within the scope of the MASA.";
         }
         leaf pinned-domain-cert {
           type binary;
           mandatory false;
           description
             "An X.509 v3 certificate structure, as specified by
              RFC 5280, using Distinguished Encoding Rules (DER)
              encoding, as defined in ITU-T X.690.

              This certificate is used by a pledge to trust a Public Key
              Infrastructure in order to verify a domain certificate
              supplied to the pledge separately by the bootstrapping
              protocol.  The domain certificate MUST have this
              certificate somewhere in its chain of certificates.
              This certificate MAY be an end-entity certificate,
              including a self-signed entity.";
           reference
             "RFC 5280:
                Internet X.509 Public Key Infrastructure Certificate
                and Certificate Revocation List (CRL) Profile.
              ITU-T X.690:
                 Information technology - ASN.1 encoding rules:
                 Specification of Basic Encoding Rules (BER),
                 Canonical Encoding Rules (CER) and Distinguished
                 Encoding Rules (DER).";
         }
         leaf domain-cert-revocation-checks {
           type boolean;
           description
```

```
             "A processing instruction to the pledge that it MUST (true)
              or MUST NOT (false) verify the revocation status for the
              pinned domain certificate.  If this field is not set, then
              normal PKIX behavior applies to validation of the domain
              certificate.";
         }
         leaf nonce {
           type binary {
             length "8..32";
           }
           must 'not(../expires-on)';
           description
             "A value that can be used by a pledge in some bootstrapping
              protocols to enable anti-replay protection.  This node is
              optional because it is not used by all bootstrapping
              protocols.

              When present, the pledge MUST compare the provided nonce
              value with another value that the pledge randomly
              generated and sent to a bootstrap server in an earlier
              bootstrapping message.  If the value is present, but
              the values do not match, then the pledge MUST NOT process
              this voucher.";
         }
         leaf pinned-domain-pubk {
           type binary;
           description
             "The pinned-domain-pubk may replace the
                pinned-domain-cert in constrained uses of
                the voucher. The pinned-domain-pubk
                is the Raw Public Key of the Registrar.
                This field is encoded as a Subject Public Key Info block
                as specified in RFC7250, in section 3.
                The ECDSA algorithm MUST be supported.
                The EdDSA algorithm as specified in
                draft-ietf-tls-rfc4492bis-17 SHOULD be supported.
                Support for the DSA algorithm is not recommended.
                Support for the RSA algorithm is a MAY.";
         }
         leaf pinned-domain-pubk-sha256 {
           type binary;
           description
             "The pinned-domain-pubk-sha256 is a second
                alternative to pinned-domain-cert.  In many cases the
                public key of the domain has already been transmitted
                during the key agreement process, and it is wasteful
                to transmit the public key another two times.
                The use of a hash of public key info, at 32-bytes for
                sha256 is a significant savings compared to an RSA
                public key, but is only a minor savings compared to
                a 256-bit ECDSA public-key.
                Algorithm agility is provided by extensions to this
                specification which can define a new leaf for another
                hash type.";
         }
         leaf last-renewal-date {
           type yang:date-and-time;
           must '../expires-on';
           description
             "The date that the MASA projects to be the last date it
              will renew a voucher on. This field is merely
              informative; it is not processed by pledges.

              Circumstances may occur after a voucher is generated that
              may alter a voucher's validity period.  For instance,
              a vendor may associate validity periods with support
              contracts, which may be terminated or extended
              over time.";
         }
         // from BRSKI-CLOUD
         leaf est-domain {
           type ietf:uri;
           description
             "The est-domain is a URL to which the Pledge should
                continue doing enrollment rather than with the
                Cloud Registrar.
                The pinned-domain-cert contains a trust-anchor
                which is to be used to authenticate the server
                found at this URI.
               ";
         }
         leaf additional-configuration {
           type ietf:uri;
           description
             "The additional-configuration attribute contains a
                URL to which the Pledge can retrieve additional
                configuration information.
                The contents of this URL are vendor specific.
                This is intended to do things like configure
                a VoIP phone to point to the correct hosted
                PBX, for example.";
         }
       } // end voucher
     } // end voucher-grouping

   }
```

<CODE ENDS>

6.4. ietf-voucher SID values

[RFC9148] explains how to serialize YANG into CBOR, and for this a series of SID values are required. While [I-D.ietf-core-sid] defines the management process for these values, due to the immaturity of the tooling around this YANG-SID mechanisms, the following values are considered normative. It is believed, however, that they will not change.

SID	Assigned to
2451	data /ietf-voucher:voucher/voucher
2452	data /ietf-voucher:voucher/voucher/assertion
2453	data /ietf-voucher:voucher/voucher/created-on
2454	data .../domain-cert-revocation-checks
2455	data /ietf-voucher:voucher/voucher/expires-on
2456	data /ietf-voucher:voucher/voucher/idevid-issuer
2457	data /ietf-voucher:voucher/voucher/last-renewal-date
2458	data /ietf-voucher:voucher/voucher/nonce
2459	data /ietf-voucher:voucher/voucher/pinned-domain-cert
2460	data /ietf-voucher:voucher/voucher/pinned-domain-pubk
2461	data .../pinned-domain-pubk-sha256
2462	data /ietf-voucher:voucher/voucher/serial-number

WARNING, obsolete definitions

The "assertion" attribute is an enumerated type [RFC8366], and the current PYANG tooling does not document the valid values for this attribute. In the JSON serialization, the literal strings from the enumerated types are used so there is no ambiguity. In the CBOR serialization, a small integer is used. This following values are documented here, but the YANG module should be considered authoritative. No IANA registry is provided or necessary because the YANG module provides for extensions.

Integer	Assertion Type
0	verified
1	logged
2	proximity
3	agent-proximity

Table 2: CBOR integers for the "assertion" attribute enum

6.5. CMS Format Voucher Artifact

The IETF evolution of PKCS#7 is CMS [RFC5652]. A CMS-signed voucher, the default type, contains a ContentInfo structure with the voucher content. An eContentType of 40 indicates that the content is a JSON-encoded voucher.

The signing structure is a CMS SignedData structure, as specified by Section 5.1 of [RFC5652], encoded using ASN.1 Distinguished Encoding Rules (DER), as specified in ITU-T X.690 [ITU-T.X690.2015].

To facilitate interoperability, Section 10.3 in this document registers the media type "application/voucher-cms+json" and the filename extension ".vcj".

The CMS structure MUST contain a 'signerInfo' structure, as described in Section 5.1 of [RFC5652], containing the signature generated over the content using a private key trusted by the recipient. Normally, the recipient is the pledge and the signer is the MASA. Another possible use could be as a "signed voucher request" format originating from the pledge or registrar toward the MASA. Within this document, the signer is assumed to be the MASA.

Note that Section 5.1 of [RFC5652] includes a discussion about how to validate a CMS object, which is really a PKCS7 object (cmsVersion=1). Intermediate systems (such the Bootstrapping Remote Secure Key Infrastructures [BRSKI] registrar) that might need to evaluate the voucher in flight MUST be prepared for such an older format. No signaling is necessary, as the manufacturer knows the capabilities of the pledge and will use an appropriate format voucher for each pledge.

The CMS structure SHOULD also contain all of the certificates leading up to and including the signer's trust anchor certificate known to the recipient. The inclusion of the trust anchor is unusual in many applications, but third parties cannot accurately audit the transaction without it.

The CMS structure MAY also contain revocation objects for any intermediate certificate authorities (CAs) between the voucher issuer and the trust anchor known to the recipient. However, the use of CRLs and other validity mechanisms is discouraged, as the pledge is unlikely to be able to perform online checks and is unlikely to have a trusted clock source. As described below, the use of short-lived vouchers and/or a pledge-provided nonce provides a freshness guarantee.

7. Voucher Request Artifact

[BRSKI], Section 3 defined a Voucher-Request Artifact as an augmented artifact from the Voucher Artifact originally defined in [RFC8366]. That definition has been moved to this document, and translated from YANG-DATA [RFC8040] to the SX:STRUCTURE extension [RFC8971].

7.1. Tree Diagram

The following tree diagram illustrates a high-level view of a voucher request document. The notation used in this diagram is described in [RFC8340]. Each node in the diagram is fully described by the YANG module in Section 7.2.

```
   module: ietf-voucher-request

     structure voucher:
       +-- voucher
          +-- created-on?
          |       yang:date-and-time
          +-- expires-on?
          |       yang:date-and-time
          +-- assertion?                                enumeration
          +-- serial-number                             string
          +-- idevid-issuer?                            binary
          +-- pinned-domain-cert?                       binary
          +-- domain-cert-revocation-checks?            boolean
          +-- nonce?                                    binary
          +-- pinned-domain-pubk?                       binary
          +-- pinned-domain-pubk-sha256?                binary
          +-- last-renewal-date?
          |       yang:date-and-time
          +-- est-domain?                               ietf:uri
          +-- additional-configuration?                 ietf:uri
          +-- prior-signed-voucher-request?             binary
          +-- proximity-registrar-cert?                 binary
          +-- proximity-registrar-pubk?                 binary
          +-- proximity-registrar-pubk-sha256?          binary
          +-- agent-signed-data?                        binary
          +-- agent-provided-proximity-registrar-cert?   binary
          +-- agent-sign-cert?                          binary
```

7.2. "ietf-voucher-request" Module

The ietf-voucher-request YANG module is derived from the ietf-voucher module.

```
   <CODE BEGINS>
   module ietf-voucher-request {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-voucher-request";
     prefix vcr;

     import ietf-yang-structure-ext {
       prefix sx;
     }
     import ietf-voucher {
       prefix vch;
       description
         "This module defines the format for a voucher,
          which is produced by a pledge's manufacturer or
          delegate (MASA) to securely assign a pledge to
```

```
            an 'owner', so that the pledge may establish a secure
            connection to the owner's network infrastructure";
         reference
           "RFC 8366: Voucher Artifact for
            Bootstrapping Protocols";
       }

       organization
         "IETF ANIMA Working Group";
       contact
         "WG Web:   <https://datatracker.ietf.org/wg/anima/>
          WG List:  <mailto:anima@ietf.org>
          Author:   Kent Watsen
                    <mailto:kent+ietf@watsen.net>
          Author:   Michael H. Behringer
                    <mailto:Michael.H.Behringer@gmail.com>
          Author:   Toerless Eckert
                    <mailto:tte+ietf@cs.fau.de>
          Author:   Max Pritikin
                    <mailto:pritikin@cisco.com>
          Author:   Michael Richardson
                    <mailto:mcr+ietf@sandelman.ca>";
       description
         "This module defines the format for a voucher request.
          It is a superset of the voucher itself.
          It provides content to the MASA for consideration
          during a voucher request.

          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 2023-01-10 {
```

```
       description
         "Initial version";
       reference
         "RFC XXXX: Bootstrapping Remote Secure Key Infrastructure";
     }

     // Top-level statement
     sx:structure voucher {
       uses voucher-request-grouping;
     }

     // Grouping defined for future usage

     grouping voucher-request-grouping {
       description
         "Grouping to allow reuse/extensions in future work.";
       uses vch:voucher-artifact-grouping {
         refine "voucher/created-on" {
           mandatory false;
         }
         refine "voucher/pinned-domain-cert" {
           mandatory false;
           description
             "A pinned-domain-cert field
              is not valid in a voucher request, and
              any occurrence MUST be ignored";
         }
         refine "voucher/last-renewal-date" {
           description
             "A last-renewal-date field
              is not valid in a voucher request, and
              any occurrence MUST be ignored";
         }
         refine "voucher/domain-cert-revocation-checks" {
           description
             "The domain-cert-revocation-checks field
              is not valid in a voucher request, and
              any occurrence MUST be ignored";
         }
         refine "voucher/assertion" {
           mandatory false;
           description
             "Any assertion included in registrar voucher
              requests SHOULD be ignored by the MASA.";
         }
         augment "voucher" {
           description
             "Adds leaf nodes appropriate for requesting vouchers.";
           leaf prior-signed-voucher-request {
             type binary;
             description
               "If it is necessary to change a voucher, or re-sign and
                forward a voucher that was previously provided along a
                protocol path, then the previously signed voucher SHOULD
                be included in this field.

                For example, a pledge might sign a voucher request
                with a proximity-registrar-cert, and the registrar
                then includes it as the prior-signed-voucher-request
                field.  This is a simple mechanism for a chain of
                trusted parties to change a voucher request, while
                maintaining the prior signature information.

                The Registrar and MASA MAY examine the prior signed
                voucher information for the
                purposes of policy decisions. For example this
                information could be useful to a MASA to determine
                that both pledge and registrar agree on proximity
                assertions. The MASA SHOULD remove all
                prior-signed-voucher-request information when
                signing a voucher for imprinting so as to minimize
                the final voucher size.";
           }
           leaf proximity-registrar-cert {
             type binary;
             description
               "An X.509 v3 certificate structure as specified by
                RFC 5280, Section 4 encoded using the ASN.1
                distinguished encoding rules (DER), as specified
                in [ITU.X690.1994].

                The first certificate in the Registrar TLS server
                certificate_list sequence  (the end-entity TLS
                certificate, see [RFC8446]) presented by the Registrar
                to the Pledge.
                This MUST be populated in a Pledge's voucher request
                when a proximity assertion is requested.";
           }
           leaf proximity-registrar-pubk {
             type binary;
             description
               "The proximity-registrar-pubk replaces
                the proximity-registrar-cert in constrained uses of
                the voucher-request.
                The proximity-registrar-pubk is the
                Raw Public Key of the Registrar. This field is encoded
                as specified in RFC7250, section 3.
                The ECDSA algorithm MUST be supported.
                The EdDSA algorithm as specified in
                draft-ietf-tls-rfc4492bis-17 SHOULD be supported.
                Support for the DSA algorithm is not recommended.
                Support for the RSA algorithm is a MAY, but due to
                size is discouraged.";
           }
           leaf proximity-registrar-pubk-sha256 {
             type binary;
             description
               "The proximity-registrar-pubk-sha256
                is an alternative to both
                proximity-registrar-pubk and pinned-domain-cert.
                In many cases the public key of the domain has already
                been transmitted during the key agreement protocol,
                and it is wasteful to transmit the public key another
                two times.
                The use of a hash of public key info, at 32-bytes for
                sha256 is a significant savings compared to an RSA
                public key, but is only a minor savings compared to
                a 256-bit ECDSA public-key.
                Algorithm agility is provided by extensions to this
                specification which may define a new leaf for another
                hash type.";
           }
           leaf agent-signed-data {
             type binary;
             description
               "The agent-signed-data field contains a JOSE [RFC7515]
                object provided by the Registrar-Agent to the Pledge.

                This artifact is signed by the Registrar-Agent
                and contains a copy of the pledge's serial-number.";
           }
           leaf agent-provided-proximity-registrar-cert {
             type binary;
             description
               "An X.509 v3 certificate structure, as specified by
                RFC 5280, Section 4, encoded using the ASN.1
                distinguished encoding rules (DER), as specified
                in ITU X.690.
                The first certificate in the registrar TLS server
                certificate_list sequence (the end-entity TLS
                certificate; see RFC 8446) presented by the
                registrar to the registrar-agent and provided to
                the pledge.
                This MUST be populated in a pledge's voucher-request
                when an agent-proximity assertion is requested.";
             reference
               "ITU X.690: Information Technology - ASN.1 encoding
                rules: Specification of Basic Encoding Rules (BER),
                Canonical Encoding Rules (CER) and Distinguished
                Encoding Rules (DER)
                RFC 5280: Internet X.509 Public Key Infrastructure
                Certificate and Certificate Revocation List (CRL)
                Profile
                RFC 8446: The Transport Layer Security (TLS)
                Protocol Version 1.3";
           }
           leaf agent-sign-cert {
             type binary;
             description
               "An X.509 v3 certificate structure, as specified by
                RFC 5280, Section 4, encoded using the ASN.1
                distinguished encoding rules (DER), as specified
                in ITU X.690.
                This certificate can be used by the pledge,
                the registrar, and the MASA to verify the signature
                of agent-signed-data. It is an optional component
                for the pledge-voucher request.
                This MUST be populated in a registrar's
                voucher-request when an agent-proximity assertion
                is requested.";
             reference
               "ITU X.690: Information Technology - ASN.1 encoding
                rules: Specification of Basic Encoding Rules (BER),
                Canonical Encoding Rules (CER) and Distinguished
                Encoding Rules (DER)
                RFC 5280: Internet X.509 Public Key Infrastructure
                Certificate and Certificate Revocation List (CRL)
                Profile";
           }
         }
       }
     }
   }
   <CODE ENDS>
```

7.3. ietf-voucher-request SID values

[RFC9148] explains how to serialize YANG into CBOR, and for this a series of SID values are required. While [I-D.ietf-core-sid] defines the management process for these values, due to the immaturity of the tooling around this YANG-SID mechanisms, the following values are considered normative. It is believed, however, that they will not change.

SID	Assigned to
2501	data /ietf-voucher-request:voucher/voucher
2515	data .../agent-provided-proximity-registrar-cert
2516	data .../agent-sign-cert
2517	data .../agent-signed-data
2502	data /ietf-voucher-request:voucher/voucher/assertion
2503	data /ietf-voucher-request:voucher/voucher/created-on
2504	data .../domain-cert-revocation-checks
2505	data /ietf-voucher-request:voucher/voucher/expires-on
2506	data .../idevid-issuer
2507	data .../last-renewal-date
2508	data /ietf-voucher-request:voucher/voucher/nonce
2509	data .../pinned-domain-cert
2518	data .../pinned-domain-pubk
2519	data .../pinned-domain-pubk-sha256
2510	data .../prior-signed-voucher-request
2511	data .../proximity-registrar-cert
2513	data .../proximity-registrar-pubk
2512	data .../proximity-registrar-pubk-sha256
2514	data .../serial-number

WARNING, obsolete definitions

The "assertion" attribute is an enumerated type, and has values as defined above in Table 2.

8. Design Considerations

8.1. Renewals Instead of Revocations

The lifetimes of vouchers may vary. In some onboarding protocols, the vouchers may be created and consumed immediately, whereas in other onboarding solutions, there may be a significant time delay between when a voucher is created and when it is consumed. In cases when there is a time delay, there is a need for the pledge to ensure that the assertions made when the voucher was created are still valid.

A revocation artifact is generally used to verify the continued validity of an assertion such as a PKIX certificate, web token, or a "voucher". With this approach, a potentially long-lived assertion is paired with a reasonably fresh revocation status check to ensure that the assertion is still valid. However, this approach increases solution complexity, as it introduces the need for additional protocols and code paths to distribute and process the revocations.

Addressing the shortcomings of revocations, this document recommends instead the use of lightweight renewals of short-lived non-revocable vouchers. That is, rather than issue a long-lived voucher, where the 'expires-on' leaf is set to some distant date, the expectation is for the MASA to instead issue a short-lived voucher, where the 'expires-on' leaf is set to a relatively near date, along with a promise (reflected in the 'last-renewal-date' field) to reissue the voucher again when needed. Importantly, while issuing the initial voucher may incur heavyweight verification checks ("Are you who you say you are?" "Does the pledge actually belong to you?"), reissuing the voucher should be a lightweight process, as it ostensibly only updates the voucher's validity period. With this approach, there is only the one artifact, and only one code path is needed to process it; there is no possibility of a pledge choosing to skip the revocation status check because, for instance, the OCSP Responder is not reachable.

While this document recommends issuing short-lived vouchers, the voucher artifact does not restrict the ability to create long-lived voucher, if required; however, no revocation method is described.

Note that a voucher may be signed by a chain of intermediate CAs leading up to the trust anchor certificate known by the pledge. Even though the voucher itself is not revocable, it may still be revoked, per se, if one of the intermediate CA certificates is revoked.

8.2. Voucher Per Pledge

The solution described herein originally enabled a single voucher to apply to many pledges, using lists of regular expressions to represent ranges of serial-numbers. However, it was determined that blocking the renewal of a voucher that applied to many devices would be excessive when only the ownership for a single pledge needed to be blocked. Thus, the voucher format now only supports a single serial-number to be listed.

9. Security Considerations

9.1. Clock Sensitivity

An attacker could use an expired voucher to gain control over a device that has no understanding of time. The device cannot trust NTP as a time reference, as an attacker could control the NTP stream.

There are three things to defend against this: 1) devices are required to verify that the expires-on field has not yet passed, 2) devices without access to time can use nonces to get ephemeral vouchers, and 3) vouchers without expiration times may be used, which will appear in the audit log, informing the security decision.

This document defines a voucher format that contains time values for expirations, which require an accurate clock in order to be processed correctly. Vendors planning on issuing vouchers with expiration values must ensure that devices have an accurate clock when shipped from manufacturing facilities and take steps to prevent clock tampering. If it is not possible to ensure clock accuracy, then vouchers with expirations should not be issued.

9.2. Protect Voucher PKI in HSM

Pursuant the recommendation made in Section 6.1 for the MASA to be deployed as an online voucher signing service, it is RECOMMENDED that the MASA's private key used for signing vouchers is protected by a hardware security module (HSM).

9.3. Test Domain Certificate Validity When Signing

If a domain certificate is compromised, then any outstanding vouchers for that domain could be used by the attacker. The domain administrator is clearly expected to initiate revocation of any domain identity certificates (as is normal in PKI solutions).

Similarly, they are expected to contact the MASA to indicate that an outstanding (presumably short lifetime) voucher should be blocked from automated renewal. Protocols for voucher distribution are RECOMMENDED to check for revocation of domain identity certificates before the signing of vouchers.

9.4. YANG Module Security Considerations

The YANG module specified in this document defines the schema for data that is subsequently encapsulated by a CMS signed-data content type, as described in Section 5 of [RFC5652]. As such, all of the YANG modeled data is protected from modification.

Implementations should be aware that the signed data is only protected from external modification; the data is still visible. This potential disclosure of information doesn't affect security so much as privacy. In particular, adversaries can glean information such as which devices belong to which organizations and which CRL Distribution Point and/or OCSP Responder URLs are accessed to validate the vouchers. When privacy is important, the CMS signed-data content type SHOULD be encrypted, either by conveying it via a mutually authenticated secure transport protocol (e.g., TLS [RFC5246]) or by encapsulating the signed-data content type with an enveloped-data content type (Section 6 of [RFC5652]), though details for how to do this are outside the scope of this document.

The use of YANG to define data structures, via the 'yang-data' statement, is relatively new and distinct from the traditional use of YANG to define an API accessed by network management protocols such as NETCONF [RFC6241] and RESTCONF [RFC8040]. For this reason, these guidelines do not follow template described by Section 3.7 of [YANG-GUIDE].

10. IANA Considerations

10.1. The IETF XML Registry

This document registers two URIs in the "IETF XML Registry" [RFC3688].

IANA has registered the following:

```
URI:  urn:ietf:params:xml:ns:yang:ietf-voucher
Registrant Contact:  The ANIMA WG of the IETF.
XML:  N/A, the requested URI is an XML namespace.
```

10.2. The YANG Module Names Registry

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

IANA is asked to registrar the following:

```
name:  ietf-voucher
namespace:  urn:ietf:params:xml:ns:yang:ietf-voucher
prefix:  vch

reference: :RFC 8366
```

IANA is asked to register a second YANG module as follows:

      name: iana-voucher-assertion-type
      namespace: urn:ietf:params:xml:ns:yang:iana-voucher-assertion-
         type
      prefix: ianavat
      reference: RFC XXXX

10.3. The Media Types Registry

This document requests IANA to update the following "Media Types" entry to point to the RFC number that will be assigned to this document:

Type name: application

Subtype name: voucher-cms+json

Required parameters: none

Optional parameters: none

Encoding considerations: CMS-signed JSON vouchers are ASN.1/DER encoded.

Security considerations: See Section 9

Interoperability considerations: The format is designed to be broadly interoperable.

Published specification: RFC 8366

Applications that use this media type: ANIMA, 6tisch, and NETCONF zero-touch imprinting systems.

Fragment identifier considerations: none

Additional information: Deprecated alias names for this type: none

Magic number(s): None

File extension(s): .vcj

Macintosh file type code(s): none

Person and email address to contact for further information: IETF AN IMA WG

Intended usage: LIMITED

Restrictions on usage: NONE

Author: ANIMA WG

Change controller: IETF

Provisional registration? (standards tree only): NO

10.4. The SMI Security for S/MIME CMS Content Type Registry

This document requests IANA to update this registered OID in the "SMI Security for S/MIME CMS Content Type (1.2.840.113549.1.9.16.1)" registry to point to the RFC number to be assigned to this document:

Decimal	Description	References
40	id-ct-animaJSONVoucher	RFC 8366

Table 3

11. References

11.1. Normative References

[BRSKI] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <https://www.rfc-editor.org/rfc/rfc8995>.

[cBRSKI] Richardson, M., Van der Stok, P., Kampanakis, P., and E. Dijk, "Constrained Bootstrapping Remote Secure Key Infrastructure (BRSKI)", Work in Progress, Internet-Draft, draft-ietf-anima-constrained-voucher-19, 2 January 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-anima-constrained-voucher-19>.

[CLOUD] Friel, O., Shekh-Yusef, R., and M. Richardson, "BRSKI Cloud Registrar", Work in Progress, Internet-Draft, draft-ietf-anima-brski-cloud-05, 13 November 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-anima-brski-cloud-05>.

[I-D.ietf-core-sid]
Veillette, M., Pelov, A., Petrov, I., Bormann, C., and M. Richardson, "YANG Schema Item iDentifier (YANG SID)", Work in Progress, Internet-Draft, draft-ietf-core-sid-19, 26 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-core-sid-19>.

[ITU-T.X690.2015]
International Telecommunication Union, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1, August 2015, <https://www.itu.int/rec/T-REC-X.690/>.

[jBRSKI] Werner, T. and M. Richardson, "JWS signed Voucher Artifacts for Bootstrapping Protocols", Work in Progress, Internet-Draft, draft-ietf-anima-jws-voucher-05, 24 October 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-anima-jws-voucher-05>.

[PRM] Fries, S., Werner, T., Lear, E., and M. Richardson, "BRSKI with Pledge in Responder Mode (BRSKI-PRM)", Work in Progress, Internet-Draft, draft-ietf-anima-brski-prm-06, 11 January 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-anima-brski-prm-06>.

[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/rfc/rfc2119>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/rfc/rfc5652>.

[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/rfc/rfc6020>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/rfc/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/rfc/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[RFC8971] Pallagatti, S., Ed., Mirsky, G., Ed., Paragiri, S., Govindan, V., and M. Mudigonda, "Bidirectional Forwarding Detection (BFD) for Virtual eXtensible Local Area Network (VXLAN)", RFC 8971, DOI 10.17487/RFC8971, December 2020, <https://www.rfc-editor.org/rfc/rfc8971>.

[RFC9148] van der Stok, P., Kampanakis, P., Richardson, M., and S. Raza, "EST-coaps: Enrollment over Secure Transport with the Secure Constrained Application Protocol", RFC 9148, DOI 10.17487/RFC9148, April 2022, <https://www.rfc-editor.org/rfc/rfc9148>.

[ZERO-TOUCH]
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/rfc/rfc8572>.

11.2. Informative References

[CBOR] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, December 2020.

<https://www.rfc-editor.org/info/std94>

[COSE] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, August 2022.

Schaad, J., "CBOR Object Signing and Encryption (COSE): Countersignatures", STD 96, RFC 9338, December 2022.

<https://www.rfc-editor.org/info/std96>

[I-D.selander-ace-ake-authz]
Selander, G., Mattsson, J. P., Vuini, M., Richardson, M., and A. Schellenbaum, "Lightweight Authorization for Authenticated Key Exchange.", Work in Progress, Internet-Draft, draft-selander-ace-ake-authz-05, 18 April 2022, <https://datatracker.ietf.org/doc/html/draft-selander-ace-ake-authz-05>.

[imprinting] Wikipedia, "Wikipedia article: Imprinting", February 2018, <https://en.wikipedia.org/w/index.php?title=Imprinting_(psychology)&oldid=825757556>.

[JWS] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/rfc/rfc7515>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/rfc/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/rfc/rfc5246>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/rfc/rfc6125>.

[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/rfc/rfc6241>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/rfc/rfc6838>.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <https://www.rfc-editor.org/rfc/rfc7435>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/rfc/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/rfc/rfc8340>.

[RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <https://www.rfc-editor.org/rfc/rfc8366>.

[SECUREJOIN] Richardson, M., "6tisch Secure Join protocol", Work in Progress, Internet-Draft, draft-ietf-6tisch-dtsecurity-secure-join-01, 25 February 2017, <https://datatracker.ietf.org/doc/html/draft-ietf-6tisch-dtsecurity-secure-join-01>.

[Stajano99theresurrecting] Stajano, F. and R. Anderson, "The Resurrecting Duckling: Security Issues for Ad-Hoc Wireless Networks", 1999, <https://www.cl.cam.ac.uk/research/dtg/www/files/publications/public/files/tr.1999.2.pdf>.

[YANG-GUIDE] 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/rfc/rfc8407>.

[YANGCBOR] Veillette, M., Ed., Petrov, I., Ed., Pelov, A., Bormann, C., and M. Richardson, "Encoding of Data Modeled with YANG in the Concise Binary Object Representation (CBOR)", RFC 9254, DOI 10.17487/RFC9254, July 2022, <https://www.rfc-editor.org/rfc/rfc9254>.

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): William Atwood, Toerless Eckert, and Sheng Jiang.

Russ Housley provided the upgrade from PKCS7 to CMS (RFC 5652) along with the detailed CMS structure diagram.

Authors’ Addresses

Kent Watsen
Watsen Networks
Email: kent+ietf@watsen.net

Michael C. Richardson
Sandelman Software

Email: mcr+ietf@sandelman.ca
URI: http://www.sandelman.ca/

Max Pritikin
Cisco Systems
Email: pritikin@cisco.com

Toerless Eckert
Futurewei Technologies Inc.
2330 Central Expy
Santa Clara, 95050
United States of America
Email: tte+ietf@cs.fau.de

Qiufang Ma
Huawei
101 Software Avenue, Yuhua District
Nanjing
210012
China
Email: maqiufang1@huawei.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 7 September 2023

R.G. Wilton
Cisco Systems
S. Mansfield
Ericsson

6 March 2023

Common Interface Extension YANG Data Models
draft-ietf-netmod-intf-ext-yang-11

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

Copyright Notice

Table of Contents

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 behavior.

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 implementers of the YANG module will decide to support all features. Hence, separate feature keywords are defined for each logically discrete feature to allow implementers 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 [RFC2119] [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 link flap suppression 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 link-flap-suppression {link-flap-suppression}?
       |  +--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}?
     augment /if:interfaces/if:interface/if:statistics:
       +--ro in-discard-unknown-encaps?   yang:counter64
               {sub-interfaces}?
```

2.1. Link Flap Suppression

The link flap suppression 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 link flap suppression 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 link flap suppression 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 link flap suppression 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 augment 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
       +--ro in-discard-overflows?            yang:counter64
```

4. Interface Extensions YANG Module

This YANG module augments the interface container defined in [RFC8343]. It also contains references to [RFC6991] and [RFC7224].

```
   <CODE BEGINS> file "ietf-if-extensions@2023-01-26.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) 2023 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 Revised 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 2023-01-26 {
       description
         "Initial revision.";

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

     feature link-flap-suppression {
       description
         "This feature indicates that configurable interface link
          delay is supported, which is a feature is used to limit the
          propagation of very short interface link state flaps.";
       reference "RFC XXXX, Section 2.1 Link Flap Suppression";
     }

     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 XXXX, Section 2.2 Dampening";
      }

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


      identity forwarding-mode {
        description "Base identity for forwarding-mode options.";
        reference "RFC XXXX, Section 2.7";
      }
      identity physical {
        base forwarding-mode;
        description
          "Physical layer forwarding.  This includes DWDM or OTN based
           optical switching.";
        reference "RFC XXXX, 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 XXXX, Section 2.7";
      }
      identity network {
        base forwarding-mode;
        description
          "Network layer based forwarding, such as IP, MPLS, or L3VPNs.";
        reference "RFC XXXX, Section 2.7";
      }


      /*
       * Augments the IETF interfaces model with leaves to configure
       * and monitor link-flap-suppression 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 Link Flap Suppression feature.
         */
        container link-flap-suppression {
          if-feature "link-flap-suppression";
          description
            "Holds link flap 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 link signal must be continuously
               down before the interface state is brought down. If not
               configured, the behavior on loss of link 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 link signal must be continuously present and error
               free before the interface state is allowed to transition
               from down to up.  If not configured, the behavior 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 link 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 link flap suppression
               timer.";
          }
          leaf timer-running {
            type enumeration {
              enum none {
                description
                  "No link flap suppression timer is running.";
              }
              enum up {
                description
                  "link-flap-suppression up timer is running.  The
                   underlying link state is up, but interface state is
                   not reported as up.";
              }
              enum down {
                description
                  "link-flap-suppression down timer is running.
                   Interface state is reported as up, but the underlying
                   link state is actually down.";
              }
            }
            config false;
            description
              "Reports whether a link flap suppression timer is actively
               running, in which case the interface state does not match
               the underlying link state.";
          }

          reference "RFC XXXX, Section 2.1 Link Flap Suppression";
        }

        /*
         * 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 XXXX, 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 XXXX, 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 XXXX, 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 XXXX, 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 XXXX, 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 XXXX, 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 XXXX, 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 XXXX, 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, 'ianaift:l2vlan') 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 XXXX, 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 XXXX, 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 XXXX, 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 XXXX, 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:l2vlan') 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 XXXX, Section 2.6 Sub-interface";
        }
      }

      /*
       * Add discard counter for unknown sub-interface encapsulation
       */
      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 interfaces that can demultiplex ingress frames to
             sub-interfaces.";
        }
        if-feature "sub-interfaces";

        description
          "Augment the interface model statistics with a sub-interface
           demux discard counter.";
        leaf in-discard-unknown-encaps {
          type yang:counter64;
          units frames;
          description
            "A count of the number of frames that were well formed, but
             otherwise discarded because their encapsulation does not
             classify the frame to the interface or any child
             sub-interface.  E.g., a frame might be discarded because the
             it has an unknown VLAN Id, or does not have a VLAN Id when
             one is expected.

             For consistency, frames counted against this counter 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>
```

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, Switch Virtual interfaces, and Pseudo-Wire Head-End interfaces. It also contains references to [RFC6991], [RFC7224], and [IEEE_802.3.2_2019].

```
   <CODE BEGINS> file "ietf-if-ethernet-like@2023-01-26.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) 2022 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 Revised 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 2023-01-26 {
       description "Initial revision.";

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

            For consistency, frames counted against this counter 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).";
       }

       leaf in-discard-overflows {
         type yang:counter64;
         units frames;
         description
```

```
            "A count of the number of frames discarded because of
             overflows.";
        }
      }
   }
   <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 link flap suppression 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:link-flap-suppression>
         <if-ext:down>0</if-ext:down>
         <if-ext:up>50</if-ext:up>
       </if-ext:link-flap-suppression>
     </interface>
   </interfaces>
```

The following example shows explicit link flap suppression 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:link-flap-suppression>
        <if-ext:down>50</if-ext:down>
        <if-ext:up>1000</if-ext:up>
      </if-ext:link-flap-suppression>
    </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. IANA Considerations

8.1. YANG Module Registrations

The following YANG modules are requested to be registered in the IANA "YANG Module Names" [RFC6020] registry:

The ietf-if-extensions module:

Name: ietf-if-extensions

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-if-extensions

Prefix: if-ext

Reference: [RFCXXXX]

The ietf-if-ethernet-like module:

Name: ietf-if-ethernet-like

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-if-ethernet-like

Prefix: ethlike

Reference: [RFCXXXX]

This document registers two URIs in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registrations have been made.

URI: urn:ietf:params:xml:ns:yang:ietf-if-extensions

Registrant Contact: The IESG.

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

URI: urn:ietf:params:xml:ns:yang:ietf-if-ethernet-like

Registrant Contact: The IESG.

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

9. 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 8341 [RFC8341] 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:

9.1. 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/link-flap-suppression/down

* /if:interfaces/if:interface/link-flap-suppression/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

9.2. 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

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>.

[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>.

[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>.

10.2. Informative References

[I-D.ietf-netmod-sub-intf-vlan-model]
Wilton, R. and S. Mansfield, "Sub-interface VLAN YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-sub-intf-vlan-model-08, 26 January 2023, <https://www.ietf.org/archive/id/draft-ietf-netmod-sub-intf-vlan-model-08.txt>.

[IEEE_802.3.2_2019]
IEEE, "IEEE Standard for Ethernet - YANG Data Model Definitions", IEEE 802-3, DOI 10.1109/IEEESTD.2019.8737019, 14 June 2019, <https://ieeexplore.ieee.org/document/8737019>.

[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>.

[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>.

[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>.

Authors' Addresses

Robert Wilton
Cisco Systems
Email: rwilton@cisco.com

Scott Mansfield
Ericsson
Email: scott.mansfield@ericsson.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 31 July 2023

R.G. Wilton, Ed.
Cisco Systems
S. Mansfield, Ed.
Ericsson

27 January 2023

Sub-interface VLAN YANG Data Models
draft-ietf-netmod-sub-intf-vlan-model-08

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. L2VPN 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 orignating 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

Copyright Notice

Table of Contents

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, for example, IPv6 [RFC8200], Ethernet Pseudo Wires [RFC4448] and VPLS [RFC4761] [RFC4762], when configured via appropriate YANG data models [RFC8344] [I-D.ietf-bess-l2vpn-yang], to interoperate 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 in the 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:

ietf-if-vlan-encapsulation.yang - Defines the model for basic classification of VLAN tagged traffic, normally to L3 packet forwarding services

ietf-if-flexible-encapsulation.yang - Defines the model for flexible classification of Ethernet/VLAN traffic, normally to L2 frame forwarding 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 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The term ’sub-interface’ is defined in section 2.6 of 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. Interface VLAN Encapsulation Model

The Interface VLAN encapsulation model provides appropriate leaves for termination of an 802.1Q VLAN tagged segment to a sub-interface (or interface) based L3 service, such as IP. It allows for termination of traffic with one or two 802.1Q VLAN tags.

The L3 service must be configured via a separate YANG data model, e.g., [RFC8344]. A short example of configuring 802.1Q VLAN sub-interfaces with IP using YANG is provided in Section 7.1.

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

```
   module: ietf-if-vlan-encapsulation
     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. Interface Flexible Encapsulation Model

The Interface Flexible Encapsulation model is designed to allow for the flexible provisioning of layer 2 services. It provides the capability to classify and demultiplex 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 frame header before the frame is handed off to the appropriate forwarding code for further handling. The forwarding instance, e.g., L2VPN, VPLS, etc., is configured using a separate YANG configuration model defined elsewhere, e.g., [I-D.ietf-bess-l2vpn-yang].

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 model also allows a flexible encapsulation and rewrite to be configured directly on an Ethernet or LAG interface without configuring separate child sub-interfaces. Ingress frames that do not match the encapsulation are dropped. Egress frames MUST conform to the encapsulation.

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.

A short example of configuring 802.1Q VLAN sub-interfaces with L2VPN using YANG is provided in Section 7.2.

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

```
module: ietf-if-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. VLAN Encapsulation YANG Module

This YANG module augments the 'encapsulation' container defined in ietf-if-extensions.yang [I-D.ietf-netmod-intf-ext-yang]. It also contains references to [RFC8343], [RFC7224], and [IEEE_802.1Q_2022].

```
<CODE BEGINS> file "ietf-if-vlan-encapsulation@2023-01-26.yang"
module ietf-if-vlan-encapsulation {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation";
  prefix if-vlan;

  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";
     }

     import ieee802-dot1q-types {
       prefix dot1q-types;
       revision-date 2022-01-19;
       reference
         "IEEE Std 802.1Q-2022: IEEE Standard for Local and
          metropolitan area networks -- Bridges and Bridged Networks";
     }

     import ietf-if-extensions {
       prefix if-ext;
       reference
         "RFC XXXX: Common Interface Extension YANG Data Models";
     }

     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 configuration to classify IEEE 802.1Q
        VLAN tagged Ethernet traffic by exactly matching the tag type
        and VLAN identifier of one or two 802.1Q VLAN tags in the frame.

        Copyright (c) 2023 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 Revised 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 2023-01-26 {
       description
         "Latest draft revision";
       reference
         "RFC XXXX: Sub-interface VLAN YANG Data Models";
     }

     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, 'ianaift:l2vlan') 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 basic 802.1Q
          VLAN tag classifications";

       case dot1q-vlan {
         container dot1q-vlan {

           description
             "Classifies 802.1Q VLAN tagged Ethernet frames to a
              sub-interface (or interface) by exactly matching the
              number of tags, tag type(s) and VLAN identifier(s).

              Only frames matching the classification configured on a
              sub-interface/interface are processed on that
              sub-interface/interface.

              Frames that do not match any sub-interface are processed
              directly on the parent interface, if it is associated with
              a forwarding instance, otherwise they are dropped.";

           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 are matched.";
             }

             description
               "Specifies the VLAN tag values to match against the
                outermost (first) 802.1Q VLAN tag in 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 802.1Q VLAN tags, the outermost
                  (first) tag in the frame MUST be specified and be of
                  S-VLAN type and the second tag in the frame must be of
                  C-VLAN tag type.";

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

             presence "Classify frames that have two 802.1Q VLAN tags.";

             description
               "Specifies the VLAN tag values to match against the
                second outermost 802.1Q VLAN tag in the frame.";

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

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

6. Flexible Encapsulation YANG Module

This YANG module augments the 'encapsulation' container defined in ietf-if-extensions.yang [I-D.ietf-netmod-intf-ext-yang]. This YANG module also augments the 'interface' list entry defined in [RFC8343]. It also contains references to [RFC7224], and [IEEE_802.1Q_2022].

```
   <CODE BEGINS> file "ietf-if-flexible-encapsulation@2023-01-26.yang"
   module ietf-if-flexible-encapsulation {
     yang-version 1.1;
     namespace
       "urn:ietf:params:xml:ns:yang:ietf-if-flexible-encapsulation";
     prefix if-flex;

     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";
     }

     import ieee802-dot1q-types {
       prefix dot1q-types;
       revision-date 2022-01-19;
       reference
         "IEEE Std 802.1Q-2022: IEEE Standard for Local and
          metropolitan area networks -- Bridges and Bridged Networks";
     }

     import ietf-if-extensions {
       prefix if-ext;
       reference
         "RFC XXXX: Common Interface Extension YANG Data Models";
     }

     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
        classification and rewrites of IEEE 802.1Q VLAN tagged Ethernet
        traffic.

        Copyright (c) 2022 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 Revised 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 2023-01-26 {
       description
         "Latest draft revision";
       reference
         "RFC XXXX: Sub-interface VLAN YANG Data Models";
     }

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

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

     grouping flexible-match {
       description
         "Represents a flexible frame classification:

          The rules for a flexible match are:
            1. Match-type: default, untagged, priority tag, or tag
               stack.
            2. Each tag in the stack of tags matches:
             a. tag type (802.1Q or 802.1ad) +
             b. tag value:
               i.   single tag
               ii.  set of tag ranges/values.
               iii. 'any' keyword";

       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.";
           }
         }

         case untagged {
           description
             "Match untagged Ethernet frames only";
           leaf untagged {
             type empty;

             description
               "Untagged match.  Matches all untagged traffic.";
           }
         }

         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 (first) 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 (first) 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 (first) 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.";
             }
           }
         }
       }
     }

     grouping dot1q-tag-rewrite {
       description
         "Flexible rewrite grouping.  Can be either be expressed
```

```
           symmetrically, or independently in the ingress and/or egress
           directions.";

        leaf pop-tags {
          type uint8 {
            range "1..2";
          }

          description
            "The number of 802.1Q VLAN tags to pop, or translate if used
             in conjunction with push-tags.

             Popped tags are the outermost tags on the frame.";
        }

        container push-tags {
          presence "802.1Q tags are pushed or translated";

          description
            "The 802.1Q tags to push on the front of the frame, or
             translate if configured 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 (first) 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 flexible-rewrite {
        description
          "Grouping for flexible rewrites of fields in the L2 header.

           Restricted to flexible 802.1Q VLAN tag rewrites, but could be
           extended to cover rewrites of other fields in the L2 header in
           future.";

        container dot1q-tag-rewrite {
          if-feature "dot1q-tag-rewrites";

          description
            "802.1Q VLAN tag rewrite.

             Translate operations are expressed as a combination of tag
             push and pop operations.  E.g., translating the outer tag is
             expressed as popping a single tag, and pushing a single tag.
             802.1Q tags that are translated SHOULD preserve the PCP and
             DEI fields unless if a different QoS behavior has been
             specified.";
          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, 'ianaift:l2vlan') 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 flexible
           match and rewrite for VLAN sub-interfaces.";

        case flexible {
          description
            "Flexible encapsulation and rewrite";

          container flexible {
            description
              "Flexible encapsulation allows for the matching of ranges
               and sets of 802.1Q VLAN Tags and performing rewrite
               operations on the VLAN tags.

               The structure is also designed to be extended to allow for
               matching/rewriting other fields within the L2 frame header
               if required.";

            container match {
              description
                "Flexibly classifies Ethernet frames to a sub-interface
                 (or interface) based on the L2 header fields.

                 Only frames matching the classification configured on a
                 sub-interface/interface are processed on that
                 sub-interface/interface.

                 Frames that do not match any sub-interface are processed
                 directly on the parent interface, if it is associated
                 with a forwarding instance, otherwise they are dropped.

                 If a frame could be classified to multiple
                 sub-interfaces then they get classified to the
                 sub-interface with the most specific match.  E.g.,
                 matching two VLAN tags in the frame is more specific
                 than matching the outermost VLAN tag, which is more
                 specific than the catch all 'default' match.";

              uses flexible-match;
            }

            container rewrite {
              if-feature "flexible-rewrites";

              description
                "L2 frame rewrite operations.

                 Rewrites allows for modifications to the L2 frame header
                 as it transits the interface/sub-interface.  Examples
                 include adding a VLAN tag, removing a VLAN tag, or
                 rewriting the VLAN Id carried in a VLAN tag.";

              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.

                       E.g., if a tag is pushed on ingress traffic, then
                       the reverse operation is a 'pop 1', that is
                       performed on traffic egressing the interface, so
                       a peer device sees a consistent L2 encapsulation
                       for both ingress and egress traffic.";
                  }
                }

                case asymmetrical {
                  if-feature "asymmetric-rewrites";

                  description
                    "Asymmetrical rewrite.

                     Rewrite operations may be specified in only a single
                     direction, or different rewrite operations may be
                     specified in each direction.";

                  container ingress {
                    uses flexible-rewrite;
                    description
                      "A rewrite operation that only applies to ingress
                       traffic.

                       Ingress rewrite operations are performed before
                       the frame is subsequently processed by the
                       forwarding operation.";
                  }

                  container egress {
                    uses flexible-rewrite;

                    description
                      "A rewrite operation that only applies to egress
                       traffic.";
                  }
                }
              }
            }

            container local-traffic-default-encaps {
              presence "A local traffic default encapsulation has been
                        specified.";

              description
                "Specifies the 802.1Q VLAN tags to use by default for
                 locally sourced traffic from the interface.

                 Used for encapsulations that match a range of VLANs (or
                 'any'), where the source VLAN Ids are otherwise
                 ambiguous.";

              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 (first) 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 (first) 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 (first)
                     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 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 two layer sub-interfaces, 'eth0.1' and 'eth0.2', both are child interfaces of the Ethernet interface 'eth0'.

'eth0.1' is configured to match traffic with two VLAN tags: an outer S-VLAN of 10 and an inner C-VLAN of 20.

'eth0.2' is configured to match traffic with a single S-VLAN tag, with VLAN Id 11.

```
<?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-ext="urn:ietf:params:xml:ns:yang:ietf-if-extensions">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
    </interface>
    <interface>
      <name>eth0.1</name>
      <type>ianaift:l2vlan</type>
      <if-ext:parent-interface>eth0</if-ext:parent-interface>
      <if-ext:encapsulation>
        <dot1q-vlan
         xmlns=
            "urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation">
          <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-ext:encapsulation>
      <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
        <forwarding>true</forwarding>
        <address>
          <ip>2001:db8:10::1</ip>
          <prefix-length>48</prefix-length>
        </address>
      </ipv6>
    </interface>
    <interface>
      <name>eth0.2</name>
      <type>ianaift:l2vlan</type>
      <if-ext:parent-interface>eth0</if-ext:parent-interface>
```

```
         <if-ext:encapsulation>
           <dot1q-vlan
            xmlns=
               "urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation">
             <outer-tag>
               <tag-type>dot1q-types:s-vlan</tag-type>
               <vlan-id>11</vlan-id>
             </outer-tag>
           </dot1q-vlan>
         </if-ext:encapsulation>
         <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
           <forwarding>true</forwarding>
           <address>
             <ip>2001:db8:11::1</ip>
             <prefix-length>48</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 remov the outer tag (S-VLAN 10) 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, with the tag removed before traffic is given to any service. Sub-interface 'eth1.0' is not currently bound to any service and hence traffic classified 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-ext="urn:ietf:params:xml:ns:yang:ietf-if-extensions">
       <interface>
         <name>eth0</name>
         <type>ianaift:ethernetCsmacd</type>
       </interface>
       <interface>
```

```
          <name>eth0.3</name>
          <type>ianaift:l2vlan</type>
          <if-ext:parent-interface>eth0</if-ext:parent-interface>
          <if-ext:encapsulation>
            <flexible xmlns=
            "urn:ietf:params:xml:ns:yang:ietf-if-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>1</pop-tags>
                  </dot1q-tag-rewrite>
                </symmetrical>
              </rewrite>
            </flexible>
          </if-ext:encapsulation>
        </interface>
        <interface>
          <name>eth1</name>
          <type>ianaift:ethernetCsmacd</type>
        </interface>
        <interface>
          <name>eth1.0</name>
          <type>ianaift:l2vlan</type>
          <if-ext:parent-interface>eth0</if-ext:parent-interface>
          <if-ext:encapsulation>
            <flexible xmlns=
            "urn:ietf:params:xml:ns:yang:ietf-if-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-ext: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>
         <!-- Does not Validate -->
         </vsi-root>
       </network-instance>
     </network-instances>
     <pseudowires
         xmlns="urn:ietf:params:xml:ns:yang:ietf-pseudowires">
       <pseudowire>
         <name>pw1</name>
           <peer-ip>2001:db8::50></peer-ip>
           <pw-id>100</pw-id>
       </pseudowire>
     </pseudowires>
   </config>
```

8. Acknowledgements

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

The authors would also like to thank Martin Bjorklund, Alex Campbell, Don Fedyk, 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. IANA Considerations

9.1. YANG Module Registrations

The following YANG modules are requested to be registered in the IANA "YANG Module Names" [RFC6020] registry:

The ietf-if-vlan-encapsulation module:

Name: ietf-if-vlan-encapsulation

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation

Prefix: if-vlan

Reference: [RFCXXXX]

The ietf-if-flexible-encapsulation module:

Name: ietf-if-flexible-encapsulation

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-if-flexible-encapsulation

Prefix: if-flex

Reference: [RFCXXXX]

This document registers two URIs in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registrations have been made.

URI: urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation

Registrant Contact: The IESG.

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

URI: urn:ietf:params:xml:ns:yang:ietf-if-flexible-encapsulation

Registrant Contact: The IESG.

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

10. 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] The NETCONF access control model [RFC8341] 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-vlan-encapsulation.yang

The nodes in the vlan encapsulation 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

10.2. ietf-if-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

11. References

11.1. Normative References

[I-D.ietf-netmod-intf-ext-yang]
Wilton, R. and S. Mansfield, "Common Interface Extension YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-intf-ext-yang-11, 26 January 2023, <https://www.ietf.org/archive/id/draft-ietf-netmod-intf-ext-yang-11.txt>.

[IEEE_802.1Q_2022]
IEEE, "IEEE Standard for Local and Metropolitan Area Networks--Bridges and Bridged Networks", IEEE 802-1q-2022, IEEE 802-1q®-2022, DOI 10.1109/IEEESTD.2022.10004498, 30 December 2022, <https://ieeexplore.ieee.org/document/10004498>.

[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>.

[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>.

[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>.

[RFC8344] Bjorklund, M., "A YANG Data Model for IP Management", RFC 8344, DOI 10.17487/RFC8344, March 2018, <https://www.rfc-editor.org/info/rfc8344>.

11.2. Informative References

[I-D.ietf-bess-l2vpn-yang]
Shah, H. C., Brissette, P., Chen, I., Hussain, I., Wen, B., and K. Tiruveedhula, "YANG Data Model for MPLS-based L2VPN", Work in Progress, Internet-Draft, draft-ietf-bess-l2vpn-yang-10, 2 July 2019, <https://www.ietf.org/archive/id/draft-ietf-bess-l2vpn-yang-10.txt>.

[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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[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>.

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

Scott Mansfield (editor)
Ericsson
Email: scott.mansfield@ericsson.com

NETMOD
Internet-Draft
Updates: RFC8342, RFC6241, RFC8526, RFC8040 (if approved)
Intended status: Standards Track
Expires: 8 July 2023

Q. Ma, Ed.
Q. Wu
C. Feng
Huawei
4 January 2023

System-defined Configuration
draft-ietf-netmod-system-config-01

Abstract

This document updates Network Management Datastore Architecture (NMDA) to define a read-only conventional configuration datastore called "system" to hold system-defined configurations. To avoid clients' explicit copy/paste of referenced system-defined configuration into the target configuration datastore (e.g., <running>), a "resolve-system" parameter is defined to allow the server acting as a "system client" to copy referenced system-defined nodes automatically. This solution enables clients manipulating the target configuration datastore (e.g., <running>) to reference nodes defined in <system>, override values of configurations defined in <system>, and configure descendant nodes of system-defined nodes.

This document updates RFC 8342, RFC 6241, RFC 8526 and RFC 8040.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Network Management Datastore Architecture (NMDA) [RFC8342] defines system configuration as the configuration that is supplied by the device itself and appears in <operational> when it is in use (Figure 2 in [RFC8342]).

However, there is a desire to enable a server to better structure and expose the system configuration. NETCONF/RESTCONF clients can benefit from a standard mechanism to retrieve what system configuration is available on a server.

In some cases, the NETCONF/RESTCONF client references a system configuration which isn't present in the target datastore (e.g., <running>), thus the configuration is considered invalid. To facilitate manipulation of the client configuration, having to copy the entire contents of the system configuration into the target datastore should be avoided or reduced when possible while ensuring that all referential integrity constraints are satisfied.

In some other cases, configuration of descendant nodes of system-defined configuration needs to be supported. For example, the system configuration contains an almost empty physical interface, while the client needs to be able to add, modify, or remove a number of descendant nodes. Some descendant nodes may not be modifiable (e.g., "name" and "type" set by the system).

This document updates NMDA [RFC8342] to define a read-only conventional configuration datastore called "system" to hold system-defined configurations. To avoid clients' explicit copy/paste of referenced system-defined configuration into the target configuration datastore (e.g., <running>), a "resolve-system" parameter has been defined to allow the server acting as a "system client" to copy referenced system-defined nodes automatically. The solution enables clients manipulating the target configuration datastore (e.g., <running>) to overlay and reference nodes defined in <system>, override values of configurations defined in <system>, and configure descendant nodes of system-defined nodes.

Conformance to this document requires NMDA servers to implement the "ietf-system-datastore" YANG module (Section 6).

1.1. Terminology

This document assumes that the reader is familiar with the contents of [RFC6241], [RFC7950], [RFC8342], [RFC8407], and [RFC8525] and uses terminologies from those documents.

The following terms are defined in this document:

System configuration: Configuration that is provided by the system itself. System configuration is present in <system> once it is created (regardless of being applied by the device), and appears in <intended> which is subject to validation. Applied system configuration also appears in <operational> with origin="system".

System configuration datastore: A configuration datastore holding the complete configuration provided by the system itself. This datastore is referred to as "<system>".

This document redefines the term "conventional configuration datastore" in Section 3 of [RFC8342] to add "system" to the list of conventional configuration datastores:

Conventional configuration datastore: One of the following set of configuration datastores: <running>, <startup>, <candidate>, <system>, and <intended>. These datastores share a common datastore schema, and protocol operations allow copying data between these datastores. The term "conventional" is chosen as a generic umbrella term for these datastores.

1.2. Requirements Language

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.

1.3. Updates to RFC 8342

This document updates RFC 8342 to define a configuration datastore called "system" to hold system configuration, it also redefines the term "conventional configuration datastore" from RFC 8342 to add "system" to the list of conventional configuration datastores. The contents of <system> datastore are read-only to clients but may change dynamically. The <system> aware client may retrieve all three types of system configuration defined in Section 2, reference nodes defined in <system>, override values of configurations defined in <system>, and configure descendant nodes of system-defined nodes.

The server will merge <running> and <system> to create <intended>. As always, system configuration will appear in <operational> with origin="system" when it is in use.

The <system> datastore makes system configuration visible to clients in order for being referenced or configurable prior to present in <operational>.

1.4. Updates to RFC 6241 and RFC 8526

This document augments <edit-config> and <edit-data> RPC operations defined in [RFC6241] and [RFC8526] respectively, with a new additional input parameter "resolve-system". The <copy-config> RPC operation defined in [RFC6241] is also augmented to support "resolve-system" parameter.

The "resolve-system" parameter is optional and has no value. When it is provided and the server detects that there is a reference to a system-defined node during the validation, the server will automatically copy the referenced system configuration into the validated datastore to make the configuration valid without the client doing so explicitly. Legacy clients interacting with servers that support this parameter don’t see any changes in <edit-config>/<edit-data> and <copy-config> behaviors.

The server’s copy referenced nodes from <system> to the target datastore MUST be enforced at the end of the <edit-config>/<edit-data> or <copy-config> operations, regardless of which target datastore it is.

1.5. Updates to RFC 8040

This document extends Sections 4.8 and 9.1.1 of [RFC8040] to add a new query parameter "resolve-system" and corresponding query parameter capability URI.

1.5.1. Query Parameter

The "resolve-system" parameter controls whether to allow a server copy any referenced system-defined configuration automatically without the client doing so explicitly. This parameter is only allowed with no values carried. If this parameter has any unexpected value, then a "400 Bad Request" status-line is returned.

Name	Methods	Description
resolve-system	POST, PUT PATCH	resolve any references not resolved by the client and copy referenced system configuration into <running> automatically. This parameter can be given in any order.

1.5.2. Query Parameter URI

To enable a RESTCONF client to discover if the "resolve-system" query parameter is supported by the server, the following capability URI is defined, which is advertised by the server if supported, using the "ietf-restconf-monitoring" module defined in RFC 8040:

urn:ietf:params:restconf:capability:resolve-system:1.0

Comment: Should we define a similar capability identifier for NETCONF protocol?

2. Kinds of System Configuration

There are three types of system configurations defined in this document: immediately-active system configuration, conditionally-active system configuration, and inactive-until-referenced system configuration.

Active system configuration refers to configuration that is in use by a device. As per definition of the operational state datastore in [RFC8342], if system configuration is inactive, it should not appear in <operational>. However, system configuration is present in <system> once it is generated, regardless of whether it is active or not.

2.1. Immediately-Active

Immediately-active system configurations are those generated in <system> and applied immediately when the device is powered on (e.g., a loopback interface), irrespective of physical resource present or not, a special functionality enabled or not.

2.2. Conditionally-Active

System configurations which are generated in <system> and applied based on specific conditions being met in a system, e.g., if a physical resource is present (e.g., insert interface card), the system will automatically detect it and load pre-provisioned configuration; when the physical resource is not present(remove interface card), the system configuration will be automatically cleared. Another example is when a special functionality is enabled, e.g., when a QoS feature is enabled, related QoS policies are automatically created by the system.

2.3. Inactive-Until-Referenced

There are some system configurations predefined (e.g., application ids, anti-x signatures, trust anchor certs, etc.) as a convenience for the clients, which must be referenced to be active. The clients can also define their own configurations for their unique requirements. Inactive-until-referenced system configurations are generated in <system> immediately when the device is powered on, but they are not applied and active until being referenced.

3. The <system> Configuration Datastore

NMDA servers compliant with this document MUST implement a <system> configuration datastore, and they SHOULD also implement the <intended> datastore.

Following guidelines for defining datastores in the appendix A of [RFC8342], this document introduces a new datastore resource named 'system' that represents the system configuration.

* Name: "system"

* YANG modules: all

* YANG nodes: all "config true" data nodes up to the root of the tree, generated by the system

* Management operations: The content of the datastore is set by the server in an implementation dependent manner. The content can not be changed by management operations via protocols such as NETCONF, RESTCONF, but may change itself by upgrades and/or when resource-conditions are met. The datastore can be read using the standard network management protocols such as NETCONF and RESCTCONF.

* Origin: This document does not define any new origin identity when it interacts with <intended> datastore and flows into <operational>. The "system" origin Metadata Annotation [RFC7952] is used to indicate the origin of a data item is system.

* Protocols: YANG-driven management protocols, such as NETCONF and RESTCONF.

* Defining YANG module: "ietf-system-datastore".

The datastore's content is defined by the server and read-only to clients. Upon the content is created or changed, it will be merged into <intended> datastore. Unlike <factory-default>[RFC8808], it MAY change dynamically, e.g., depending on factors like device upgrade or system-controlled resources change (e.g., HW available). The <system> datastore doesn't persist across reboots; the contents of <system> will be lost upon reboot and recreated by the system with the same or changed contents. <factory-reset> RPC operation defined in [RFC8808] can reset it to its factory default configuration without including configuration generated due to the system update or client-enabled functionality.

The <system> datastore is defined as a conventional configuration datastore and shares a common datastore schema with other conventional datastores. The <system> configuration datastore must always be valid, as defined in Section 8.1 of [RFC7950].

4. Static Characteristics of the <system> Configuration Datastore

4.1. Read-only to Clients

The <system> configuration datastore is a read-only configuration datastore (i.e., edits towards <system> directly MUST be denied), though the client may be allowed to override the value of a system-initialized data node (see Section 5.4).

4.2. May Change via Software Upgrades

System configuration may change dynamically, e.g., depending on factors like device upgrade or if system-controlled resources (e.g., HW available) change. In some implementations, when a QoS feature is enabled, QoS-related policies are created by the system. If the system configuration gets changed, YANG notifications (e.g., "push-change-update" notification) [RFC6470][RFC8639][RFC8641] can be used to notify the client. Any update of the contents in <system> will not cause the automatic update of <running>, even if some of the system configuration has already been copied into <running> explicitly or automatically before the update.

4.3. No Impact to <operational>

This work intends to have no impact to <operational>. System configuration will appear in <operational> with "origin=system". This document enables a subset of those system generated nodes to be defined like configuration, i.e., made visible to clients in order for being referenced or configurable prior to present in <operational>. "Config false" nodes are out of scope, hence existing "config false" nodes are not impacted by this work.

5. Dynamic Behavior

5.1. Conceptual Model of Datastores

This document introduces a datastore named "system" which is used to hold all three types of system configurations defined in Section 2.

When the device is powered on, immediately-active system configuration will be generated in <system> and applied immediately but inactive-until-referenced system configuration only becomes active if it is referenced by client-defined configuration. While conditionally-active system configuration will be created and immediately applied if the condition on system resources is met when the device is powered on or running.

All above three types of system configurations will appear in <system>. Clients MAY reference nodes defined in <system>, override values of configurations defined in <system>, and configure descendant nodes of system-defined nodes, by copying or writing intended configurations into the target configuration datastore (e.g., <running>).

The server will merge <running> and <system> to create <intended>, in which process, the data node appears in <running> takes precedence over the same node in <system> if the server allows the node to be modifiable; additional nodes to a list entry or new list/leaf-list entries appear in <running> extends the list entry or the whole list/leaf-list defined in <system> if the server allows the list/leaf-list to be updated. In addition, the <intended> configuration datastore represents the configuration after all configuration transformation to <system> are performed (e.g., system-defined template expansion, removal of inactive system configuration). If a server implements <intended>, <system> MUST be merged into <intended>.

Servers MUST enforce that configuration references in <running> are resolved within the <running> datastore and ensure that <running> contains any referenced system configuration. Clients MUST either explicitly copy system-defined nodes into <running> or use the "resolve-system" parameter. The server MUST enforce that the referenced system nodes configured into <running> by the client is consistent with <system>. Note that <system> aware clients know how to discover what nodes exist in <system>. How clients unaware of the <system> datastore can find appropriate configurations is beyond the scope of this document.

No matter how the referenced system configurations are copied into <running>, the nodes copied into <running> would always be returned after a read of <running>, regardless if the client is <system> aware.

Configuration defined in <system> is present in <operational> if it is actively in use by the device, even if a client may delete the configuration copied from <system> into <running>. For example, system initializes a value for a particular leaf which is overridden by the client with a different value in <running>. The client may delete that node in <running>, in which case system-initialized value defined in <system> can be still in use and appear in <operational>. Any deletable system-provided configuration must be defined in <factory-default> [RFC8808], which is used to initialize <running> when the device is first-time powered on or reset to its factory default condition.

5.2. Explicit Declaration of System Configuration

It is possible for a client to explicitly declare system configuration nodes in the target datastore (e.g., <running>) with the same values as in <system>, by configuring a node (list/leaf-list entry, leaf, etc.) in the target datastore (e.g., <running>) that matches the same node and value in <system>.

This explicit configuration of system-defined nodes in <running> can be useful, for example, when the client doesn't want a "system client" to have a role or hasn't implemented the "resolve-system" parameter. The client can explicitly declare (i.e., configure in <running>) the list entries (with at least the keys) for any system configuration list entries that are referenced elsewhere in <running>. The client does not necessarily need to declare all the contents of the list entry (i.e. the descendant nodes) , only the parts that are required to make the <running> appear valid.

5.3. Servers Auto-configuring Referenced System Configuration

This document defines a new parameter "resolve-system" to the input for the <edit-config>, <edit-data>, and <copy-config> operations. Clients that are aware of the "resolve-system" parameter MAY use this parameter to avoid the requirement to provide a referentially complete configuration in <running>.

Non-NMDA servers MAY implement this parameter without implementing the <system> configuration datastore, which would only eliminate the ability to expose the system configuration via protocol operations. If a server implements <system>, referenced system configuration is copied from <system> into the target datastore(e.g., <running>) when the "resolve-system" parameter is used; otherwise it is an implementation decision where to copy referenced system configuration into the target datastore(e.g., <running>).

If the "resolve-system" is present, and the server supports this capability, the server MUST copy relevant referenced system-defined nodes into the target datastore (e.g., <running>) without the client doing the copy/paste explicitly, to resolve any references not resolved by the client. The server acting as a "system client" like any other remote clients copies the referenced system-defined nodes when triggered by the "resolve-system" parameter.

If the "resolve-system" parameter is not given by the client, the server should not modify <running> in any way otherwise not specified by the client. Not using capitalized "SHOULD NOT" in the previous sentence is intentional. The intention is to bring awareness to the general need to not surprise clients with unexpected changes. It is desirable for clients to always opt into using mechanisms having server-side changes. This document enables a client to opt into this behavior using the "resolve-system" parameter. RFC 7317 enables a client to opt into its behavior using a "$0$" prefix (see ianach:crypt-hash type defined in [RFC7317]).

The server may automatically configure the list entries (with at least the keys) in the target datastore (e.g., <running>) for any system configuration list entries that are referenced elsewhere by the clients. Similarly, not all the contents of the list entry (i.e., the descendant nodes) are necessarily copied by the server - only the parts that are required to make the <running> valid. A read back of <running> (i.e., <get>, <get-config> or <get-data> operation) returns those automatically copied nodes.

5.4. Modifying (overriding) System Configuration

In some cases, a server may allow some parts of system configuration to be modified. Modification of system configuration is achieved by the client writing configuration to <running> that overrides the system configuration. Configurations defined in <running> take precedence over system configuration nodes in <system> if the server allows the nodes to be modified.

For instance, list keys in system configuration can't be changed by a client, but other descendant nodes in a list entry may be modifiable or non-modifiable. Leafs and leaf-lists outside of lists may also be modifiable or non-modifiable. Even if some system configuration has been copied into <running> earlier, whether it is modifiable or not in <running> follows general YANG constraints and NACM rules, and other server-internal restrictions. If a system configuration node is non-modifiable, then writing a different value for that node MUST return an error. The immutability of system configuration is further defined in [I-D.ma-netmod-immutable-flag].

A server may also allow a client to add data nodes to a list entry in <system> by writing those additional nodes in <running>. Those additional data nodes may not exist in <system> (i.e., an *addition* rather than an override).

Comment 1: What if <system> contains a set of values for a leaf-list, and a client configures another set of values for that leaf-list in <running>, will the set of values in <running> completely replace the set of values in <system>? Or the two sets of values are merged together?

Comment 2: how "ordered-by user" lists and leaf-lists are merged? Do the <running> values go before or after, or is this a case where a full-replace is needed.

5.5. Examples

This section shows some examples of server-configuring of <running> automatically, declaring a system-defined node in <running> explicitly, modifying a system-instantiated leaf's value and configuring descendant nodes of a system-defined node. For each example, the corresponding XML snippets are provided.

5.5.1. Server Configuring of <running> Automatically

In this subsection, the following fictional module is used:

```
module example-application {
  yang-version 1.1;
  namespace "urn:example:application";
  prefix "app";

  import ietf-inet-types {
    prefix "inet";
  }
  container applications {
    list application {
      key "name";
      leaf name {
        type string;
      }
      leaf protocol {
        type enumeration {
          enum tcp;
          enum udp;
        }
      }
      leaf destination-port {
        type inet:port-number;
      }
    }
  }
}
```

The server may predefine some applications as a convenience for the clients. These predefined configurations are applied only after being referenced by other configurations, which fall into the "inactive-until-referenced" system configuration as defined in Section 2. The system-instantiated application entries may be present in <system> as follows:

```
        <applications xmlns="urn:example:application">
          <application>
            <name>ftp</name>
            <protocol>tcp</protocol>
            <destination-port>21</destination-port>
          </application>
          <application>
            <name>tftp</name>
            <protocol>udp</protocol>
            <destination-port>69</destination-port>
          </application>
          <application>
            <name>smtp</name>
            <protocol>tcp</protocol>
            <destination-port>25</destination-port>
          </application>
          ...
        </applications>
```

The client may also define its customized applications. Suppose the configuration of applications is present in <running> as follows:

```
        <applications xmlns="urn:example:application">
          <application>
            <name>my-app-1</name>
            <protocol>tcp</protocol>
            <destination-port>2345</destination-port>
          </application>
          <application>
            <name>my-app-2</name>
            <protocol>udp</protocol>
            <destination-port>69</destination-port>
          </application>
        </applications>
```

A fictional ACL YANG module is used as follows, which defines a leafref for the leaf-list "application" data node to refer to an existing application name.

```
         module example-acl {
           yang-version 1.1;
           namespace "urn:example:acl";
           prefix "acl";

           import example-application {
             prefix "app";
           }
           import ietf-inet-types {
             prefix "inet";
           }

           container acl {
             list acl_rule {
               key "name";
               leaf name {
                 type string;
               }
               container matches {
                 choice l3 {
                   container ipv4 {
                     leaf source_address {
                       type inet:ipv4-prefix;
                     }
                     leaf dest_address {
                       type inet:ipv4-prefix;
                     }
                   }
                 }
                 choice applications {
                   leaf-list application {
                     type leafref {
                     path "/app:applications/app:application/app:name";
                     }
                   }
                 }
               }
               leaf packet_action {
                 type enumeration {
                   enum forward;
                   enum drop;
                   enum redirect;
                 }
               }
             }
           }
         }
```

If a client configures an ACL rule referencing system predefined nodes which are not present in <running>, the client may issue an <edit-config> operation with the parameter "resolve-system" as follows:

```
       <rpc message-id="101"
            xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
         <edit-config>
           <target>
             <running/>
           </target>
           <config>
             <acl xmlns="urn:example:acl">
               <acl_rule>
                 <name>allow_access_to_ftp_tftp</name>
                 <matches>
                   <ipv4>
                     <source_address>198.51.100.0/24</source_address>
                     <dest_address>192.0.2.0/24</dest_address>
                   </ipv4>
                   <application>ftp</application>
                   <application>tftp</application>
                   <application>my-app-1</application>
                 </matches>
                 <packet_action>forward</packet_action>
               </acl_rule>
             </acl>
           </config>
           <resolve-system/>
         </edit-config>
       </rpc>
```

Then following gives the configuration of applications in <running> which is returned in the response to a follow-up <get-config> operation:

```
           <applications xmlns="urn:example:application">
             <application>
               <name>my-app-1</name>
               <protocol>tcp</protocol>
               <destination-port>2345</destination-port>
             </application>
             <application>
               <name>my-app-2</name>
               <protocol>udp</protocol>
               <destination-port>69</destination-port>
             </application>
             <application>
               <name>ftp</name>
             </application>
             <application>
               <name>tftp</name>
             </application>
           </applications>
```

Then the configuration of applications is present in <operational> as follows:

```
        <applications xmlns="urn:example:application"
                      xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                      or:origin="or:intended">
          <application>
            <name>my-app-1</name>
            <protocol>tcp</protocol>
            <destination-port>2345</destination-port>
          </application>
          <application>
            <name>my-app-2</name>
            <protocol>udp</protocol>
            <destination-port>69</destination-port>
          </application>
          <application or:origin="or:system">
            <name>ftp</name>
            <protocol>tcp</protocol>
            <destination-port>21</destination-port>
          </application>
          <application or:origin="or:system">
            <name>tftp</name>
            <protocol>udp</protocol>
            <destination-port>69</destination-port>
          </application>
        </applications>
```

Since the configuration of application "smtp" is not referenced by the client, it does not appear in <operational> but only in <system>.

5.5.2. Declaring a System-defined Node in <running> Explicitly

It’s also possible for a client to explicitly declare the system-defined configurations that are referenced. For instance, in the above example, the client MAY also explicitly configure the following system defined applications "ftp" and "tftp" only with the list key "name" before referencing:

```
          <rpc message-id="101"
               xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
             <edit-config>
               <target>
                 <running/>
               </target>
               <config>
                 <applications xmlns="urn:example:application">
                   <application>
                     <name>ftp</name>
                   </application>
                   <application>
                     <name>tftp</name>
                   </application>
                 </applications>
               </config>
             </edit-config>
           </rpc>
```

Then the client issues an <edit-config> operation to configure an ACL rule referencing applications "ftp" and "tftp" without the parameter "resolve-system" as follows:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <acl xmlns="urn:example:acl">
        <acl_rule>
          <name>allow_access_to_ftp_tftp</name>
          <matches>
            <ipv4>
              <source_address>198.51.100.0/24</source_address>
              <dest_address>192.0.2.0/24</dest_address>
            </ipv4>
            <application>ftp</application>
            <application>tftp</application>
            <application>my-app-1</application>
          </matches>
          <packet_action>forward</packet_action>
        </acl_rule>
      </acl>
    </config>
  </edit-config>
</rpc>
```

Then following gives the configuration of applications in <running> which is returned in the response to a follow-up <get-config> operation, all the configuration of applications are explicitly configured by the client:

```
           <applications xmlns="urn:example:application">
             <application>
               <name>my-app-1</name>
               <protocol>tcp</protocol>
               <destination-port>2345</destination-port>
             </application>
             <application>
               <name>my-app-2</name>
               <protocol>udp</protocol>
               <destination-port>69</destination-port>
             </application>
             <application>
               <name>ftp</name>
             </application>
             <application>
               <name>tftp</name>
             </application>
           </applications>
```

Then the configuration of applications is present in <operational> as follows:

```
        <applications xmlns="urn:example:application"
                      xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                      or:origin="or:intended">
          <application>
            <name>my-app-1</name>
            <protocol>tcp</protocol>
            <destination-port>2345</destination-port>
          </application>
          <application>
            <name>my-app-2</name>
            <protocol>udp</protocol>
            <destination-port>69</destination-port>
          </application>
          <application>
            <name>ftp</name>
            <protocol or:origin="or:system">tcp</protocol>
            <destination-port or:origin="or:system">21</destination-port>
          </application>
          <application>
            <name>tftp</name>
            <protocol or:origin="or:system">udp</protocol>
            <destination-port or:origin="or:system">69</destination-port>
          </application>
        </applications>
```

Since the application names "ftp" and "tftp" are explicitly configured by the client, they take precedence over the values in <system>, the "origin" attribute will be set to "intended".

5.5.3. Modifying a System-instantiated Leaf's Value

In this subsection, we will use this fictional QoS data model:

```
       module example-qos-policy {
         yang-version 1.1;
         namespace "urn:example:qos";
         prefix "qos";

         container qos-policies {
            list policy {
              key "name";
              leaf name {
              type string;
            }
              list queue {
                key "queue-id";
                  leaf queue-id {
                    type int32 {
                      range "1..32";
                    }
                  }
                  leaf maximum-burst-size {
                    type int32 {
                      range "0..100";
                    }
                  }
                }
              }
            }
          }
```

Suppose a client creates a qos policy "my-policy" with 4 system instantiated queues(1~4). The configuration of qos-policies is present in <system> as follows:

```
        <qos-policies xmlns="urn:example:qos">
          <name>my-policy</name>
          <queue>
            <queue-id>1</queue-id>
            <maximum-burst-size>50</maximum-burst-size>
          </queue>
          <queue>
            <queue-id>2</queue-id>
            <maximum-burst-size>60</maximum-burst-size>
          </queue>
          <queue>
            <queue-id>3</queue-id>
            <maximum-burst-size>70</maximum-burst-size>
          </queue>
          <queue>
            <queue-id>4</queue-id>
            <maximum-burst-size>80</maximum-burst-size>
          </queue>
        </qos-policies>
```

A client modifies the value of maximum-burst-size to 55 in queue-id 1:

```
        <rpc message-id="101"
             xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
          <edit-config>
            <target>
              <running/>
            </target>
            <config>
              <qos-policies xmlns="urn:example:qos">
                <name>my-policy</name>
                <queue>
                  <queue-id>1</queue-id>
                  <maximum-burst-size>55</maximum-burst-size>
                </queue>
              </qos-policies>
            </config>
          </edit-config>
        </rpc>
```

Then, the configuration of qos-policies is present in <operational> as follows:

```
        <qos-policies  xmlns="urn:example:qos"
                       xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                       or:origin="or:intended">
          <name>my-policy</name>
          <queue>
            <queue-id>1</queue-id>
            <maximum-burst-size>55</maximum-burst-size>
          </queue>
          <queue or:origin="or:system">
            <queue-id>2</queue-id>
            <maximum-burst-size>60</maximum-burst-size>
          </queue>
           <queue or:origin="or:system">
            <queue-id>3</queue-id>
            <maximum-burst-size>70</maximum-burst-size>
          </queue>
           <queue or:origin="or:system">
            <queue-id>4</queue-id>
            <maximum-burst-size>80</maximum-burst-size>
          </queue>
        </qos-policies>
```

5.5.4. Configuring Descendant Nodes of a System-defined Node

This subsection also uses the fictional interface YANG module defined in Appendix C.3 of [RFC8342]. Suppose the system provides a loopback interface (named "lo0") with a default IPv4 address of "127.0.0.1" and a default IPv6 address of "::1".

The configuration of "lo0" interface is present in <system> as follows:

```
         <interfaces>
           <interface>
             <name>lo0</name>
             <ip-address>127.0.0.1</ip-address>
             <ip-address>::1</ip-address>
           </interface>
         </interfaces>
```

The configuration of "lo0" interface is present in <operational> as follows:

```
        <interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                    or:origin="or:system">
          <interface>
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
        </interfaces>
```

Later on, the client further configures the description node of a "lo0" interface as follows:

```
        <rpc message-id="101"
             xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
          <edit-config>
            <target>
              <running/>
            </target>
            <config>
              <interfaces>
                <interface>
                  <name>lo0</name>
                  <description>loopback</description>
                </interface>
              </interfaces>
            </config>
          </edit-config>
        </rpc>
```

Then the configuration of interface "lo0" is present in <operational> as follows:

```
          <interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                      or:origin="or:intended">
            <interface>
              <name>lo0</name>
              <description>loopback</description>
              <ip-address or:origin="or:system">127.0.0.1</ip-address>
              <ip-address or:origin="or:system">::1</ip-address>
            </interface>
          </interfaces>
```

6. The "ietf-system-datastore" Module

6.1. Data Model Overview

This YANG module defines a new YANG identity named "system" that uses the "ds:datastore" identity defined in [RFC8342]. A client can discover the <system> datastore support on the server by reading the YANG library information from the operational state datastore. Note that no new origin identity is defined in this document, the "or:system" origin Metadata Annotation [RFC7952] is used to indicate the origin of a data item is system. Support for the "origin" annotation is identified with the feature "origin" defined in [RFC8526].

The following diagram illustrates the relationship amongst the "identity" statements defined in the "ietf-system-datastore" and "ietf-datastores" YANG modules:

```
Identities:
    +--- datastore
    |   +--- conventional
    |   |   +--- running
    |   |   +--- candidate
    |   |   +--- startup
    |   |   +--- system
    |   |   +--- intended
    |   +--- dynamic
    |   +--- operational
```

The diagram above uses syntax that is similar to but not defined in [RFC8340].

6.2. Example Usage

This section gives an example of data retrieval from <system>. The YANG module used are shown in Appendix C.2 of [RFC8342]. All the messages are presented in a protocol-independent manner. JSON is used only for its conciseness.

Suppose the following data is added to <running>:

```
{
    "bgp": {
        "local-as": "64501",
        "peer-as": "64502",
        "peer": {
            "name": "2001:db8::2:3"
        }
    }
}
```

REQUEST (a <get-data> or GET request sent from the NETCONF or RESTCONF client):

```
Datastore: <system>
Target:/bgp
```

An example of RESTCONF request:

```
      GET /restconf/ds/system/bgp HTTP/1.1
      Host: example.com
      Accept: application/yang-data+xml
```

RESPONSE ("local-port" leaf value is supplied by the system):

```
{
    "bgp": {
        "peer": {
            "name": "2001:db8::2:3",
            "local-port": "60794"
        }
    }
}
```

6.3. YANG Module

```
<CODE BEGINS> file "ietf-system-datastore@2023-01-05.yang"

module ietf-system-datastore {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-system-datastore";
  prefix sysds;

  import ietf-datastores {
    prefix ds;
    reference
      "RFC 8342: Network Management Datastore Architecture(NMDA)";
  }

  organization
    "IETF NETDOD (Network Modeling) Working Group";
  contact
    "WG Web:   https://datatracker.ietf.org/wg/netmod/
     WG List:  NETMOD WG list <mailto:netmod@ietf.org>

     Author: Qiufang Ma
             <mailto:maqiufang1@huawei.com>
     Author: Qin Wu
             <mailto:bill.wu@huawei.com>
```

```
        Author: Chong Feng
                <mailto:frank.fengchong@huawei.com>";
     description
       "This module defines a new YANG identity that uses the
        ds:datastore identity defined in [RFC8342].

        Copyright (c) 2022 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 Revised
        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 HHHH
        (https://www.rfc-editor.org/info/rfcHHHH); 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 2023-01-05 {
       description
         "Initial version.";
       reference
         "RFC XXXX: System-defined Configuration";
     }

     identity system {
       base ds:conventional;
       description
         "This read-only datastore contains the configuration
          provided by the system itself.";
     }
   }

   <CODE ENDS>
```

7. The "ietf-netconf-resolve-system" Module

This YANG module is optional to implement.

7.1. Data Model Overview

This YANG module augments NETCONF <edit-config>, <edit-data> and <copy-config> operations with a new parameter "resolve-system" in the input parameters. If the "resolve-system" parameter is present, the server will copy the referenced system configuration into target datastore automatically. A NETCONF client can discover the "resolve-system" parameter support on the server by checking the YANG library information with "ietf-netconf-resolve-system" YANG module included from the operational state datastore.

The following tree diagram [RFC8340] illustrates the "ietf-netconf-resolve-system" module:

```
   module: ietf-netconf-resolve-system
     augment /nc:edit-config/nc:input:
       +---w resolve-system?   empty
     augment /nc:copy-config/nc:input:
       +---w resolve-system?   empty
     augment /ncds:edit-data/ncds:input:
       +---w resolve-system?   empty
```

The following tree diagram [RFC8340] illustrates "edit-config", "copy-config" and "edit-data" rpcs defined in "ietf-netconf" and "ietf-netconf-nmda" respectively, augmented by "ietf-netconf-resolve-system" YANG module:

```
     rpcs:
       +---x edit-config
       |  +---w input
       |     +---w target
       |     |  +---w (config-target)
       |     |     +--:(candidate)
       |     |     |  +---w candidate?   empty {candidate}?
       |     |     +--:(running)
       |     |        +---w running?     empty {writable-running}?
       |     +---w default-operation?   enumeration
       |     +---w test-option?         enumeration {validate}?
       |     +---w error-option?        enumeration
       |     +---w (edit-content)
       |     |   +--:(config)
       |     |   |  +---w config?        <anyxml>
       |     |   +--:(url)
       |     |     +---w url?           inet:uri {url}?
       |     +---w resolve-system?      empty
       +---x copy-config
       |  +---w input
       |     +---w target
       |     |  +---w (config-target)
       |     |     +--:(candidate)
       |     |     |  +---w candidate?   empty {candidate}?
       |     |     +--:(running)
       |     |     |  +---w running?     empty {writable-running}?
       |     |     +--:(startup)
       |     |     |  +---w startup?     empty {startup}?
       |     |     +--:(url)
       |     |        +---w url?         inet:uri {url}?
       |     +---w source
       |     |  +---w (config-source)
       |     |     +--:(candidate)
       |     |     |  +---w candidate?   empty {candidate}?
       |     |     +--:(running)
       |     |     |  +---w running?     empty
       |     |     +--:(startup)
       |     |     |  +---w startup?     empty {startup}?
       |     |     +--:(url)
       |     |     |  +---w url?         inet:uri {url}?
       |     |     +--:(config)
       |     |        +---w config?      <anyxml>
       |     +---w resolve-system?       empty
       +---x edit-data
          +---w input
             +---w datastore            ds:datastore-ref
             +---w default-operation?   enumeration
             +---w (edit-content)
             |  +--:(config)
             |  |  +---w config?        <anydata>
             |  +--:(url)
             |     +---w url?           inet:uri {nc:url}?
             +---w resolve-system?      empty
```

7.2. Example Usage

This section gives an example of an <edit-config> request to reference system-defined data nodes which are not present in <running> with a "resolve-system" parameter. A retrieval of <running> to show the auto-copied referenced system configurations after the <edit-config> request is also given. The YANG module used is shown as follows, leafrefs refer to an existing name and address of an interface:

```
    module example-interface-management {
      yang-version 1.1;
      namespace "urn:example:interfacemgmt";
      prefix "inm";

      container interfaces {
        list interface {
          key name;
          leaf name {
            type string;
          }
          leaf description {
            type string;
          }
          leaf mtu {
            type uint16;
          }
          leaf ip-address {
            type inet:ip-address;
          }
        }
      }
      container default-address {
        leaf ifname {
          type leafref {
            path "../../interfaces/interface/name";
          }
        }
        leaf address {
          type leafref {
            path "../../interfaces/interface[name = current()/../ifname]"
               + "/ip-address";
          }
        }
      }
    }
```

Image that the system provides a loopback interface (named "lo0") with a predefined MTU value of "1500" and a predefined IP address of "127.0.0.1". The <system> datastore shows the following configuration of loopback interface:

```
   <interfaces xmlns="urn:example:interfacemgmt">
     <interface>
       <name>lo0</name>
       <mtu>1500</mtu>
       <ip-address>127.0.0.1</ip-address>
     </interface>
   </interfaces>
```

The client sends an <edit-config> operation to add the configuration of default-address with a "resolve-system" parameter:

```
  <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
    <edit-config>
      <target>
        <running/>
      </target>
      <config>
        <default-address xmlns="urn:example:interfacemgmt">
          <if-name>lo0</if-name>
          <address>127.0.0.1</address>
        </default-address>
      </config>
     <resolve-system/>
    </edit-config>
  </rpc>
```

Since the "resolve-system" parameter is provided, the server will resolve any leafrefs to system configurations and copy the referenced system-defined nodes into <running> automatically with the same value (i.e., the name and ip-address data nodes of lo0 interface) in <system> at the end of <edit-config> operation constraint enforcement. After the processing, a positive response is returned:

```
   <rpc-reply message-id="101"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <ok/>
   </rpc-reply>
```

Then the client sends a <get-config> operation towards <running>:

```
   <rpc message-id="101"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <get-config>
       <source>
         <running/>
       </source>
       <filter type="subtree">
         <interfaces xmlns="urn:example:interfacemgmt"/>
       </filter>
     </get-config>
   </rpc>
```

Given that the referenced interface "name" and "ip-address" of lo0 are configured by the server, the following response is returned:

```
   <rpc-reply message-id="101"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <data>
       <interfaces xmlns="urn:example:interfacemgmt">
         <interface>
           <name>lo0</name>
           <ip-address>127.0.0.1</ip-address>
         </interface>
       </interfaces>
     </data>
   </rpc-reply>
```

7.3. YANG Module

```
   <CODE BEGINS> file "ietf-netconf-resolve-system@2023-01-05.yang"

  =============== NOTE: '\' line wrapping per RFC 8792 ================

  module ietf-netconf-resolve-system {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-resolve-system";
    prefix ncrs;

    import ietf-netconf {
      prefix nc;
      reference
        "RFC 6241: Network Configuration Protocol (NETCONF)";
    }
    import ietf-netconf-nmda {
      prefix ncds;
      reference
        "RFC 8526: NETCONF Extensions to Support the Network
         Management Datastore Architecture";
```

```
     }

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

        Author: Qiufang Ma
                <mailto:maqiufang1@huawei.com>
        Author: Qin Wu
                <mailto:bill.wu@huawei.com>
        Author: Chong Feng
                <mailto:frank.fengchong@huawei.com>";
     description
       "This module defines an extension to the NETCONF protocol
        that allows the NETCONF client to control whether the server
        is allowed to copy referenced system configuration
        automatically without the client doing so explicitly.

         Copyright (c) 2022 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 Revised
         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 HHHH
         (https://www.rfc-editor.org/info/rfcHHHH); 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 2023-01-05 {
       description
         "Initial version.";
       reference
         "RFC XXXX: System-defined Configuration";
     }
```

```
    grouping resolve-system-grouping {
      description
        "Define the resolve-system parameter grouping.";
      leaf resolve-system {
        type empty;
        description
          "When present, the server is allowed to automatically
           configure referenced system configuration into the
           target configuration datastore.";
      }
    }

    augment "/nc:edit-config/nc:input" {
      description
        "Allows the server to automatically configure
         referenced system configuration to make configuration
         valid.";
      uses resolve-system-grouping;
    }

    augment "/nc:copy-config/nc:input" {
      description
        "Allows the server to automatically configure
         referenced system configuration to make configuration
         valid.";
      uses resolve-system-grouping;
    }

    augment "/ncds:edit-data/ncds:input" {
      description
        "Allows the server to automatically configure
         referenced system configuration to make configuration
         valid.";
      uses resolve-system-grouping;
    }
  }

   <CODE ENDS>
```

8. IANA Considerations

8.1. The "IETF XML" Registry

This document registers two XML namespace URNs in the ’IETF XML registry’, following the format defined in [RFC3688].

```
      URI: urn:ietf:params:xml:ns:yang:ietf-system-datastore
      Registrant Contact: The IESG.
      XML: N/A, the requested URIs are XML namespaces.

      URI: urn:ietf:params:xml:ns:yang:ietf-netconf-resolve-system
      Registrant Contact: The IESG.
      XML: N/A, the requested URIs are XML namespaces.
```

8.2. The "YANG Module Names" Registry

This document registers two module names in the 'YANG Module Names' registry, defined in [RFC6020] .

```
      name: ietf-system-datastore
      prefix: sys
      namespace: urn:ietf:params:xml:ns:yang:ietf-system-datatstore
      maintained by IANA: N
      RFC: XXXX // RFC Ed.: replace XXXX and remove this comment


      name: ietf-netconf-resolve-system
      prefix: ncrs
      namespace: urn:ietf:params:xml:ns:yang:ietf-netconf-resolve-system
      maintained by IANA: N
      RFC: XXXX // RFC Ed.: replace XXXX and remove this comment
```

8.3. RESTCONF Capability URN Registry

This document registers a capability in the "RESTCONF Capability URNs" registry [RFC8040]:

Index	Capability Identifier
:resolve-system	urn:ietf:params:restconf:capability:resolve-system:1.0

9. Security Considerations

9.1. Regarding the "ietf-system-datastore" YANG Module

The YANG module defined in this document extends the base operations for NETCONF [RFC6241] and 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 Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF users to a preconfigured subset of all available NETCONF protocol operations and content.

9.2. Regarding the "ietf-netconf-resolve-system" YANG Module

The YANG module defined in this document extends the base operations for NETCONF [RFC6241] and [RFC8526]. 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 Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF users to a preconfigured subset of all available NETCONF protocol operations and content.

The security considerations for the base NETCONF protocol operations (see Section 9 of [RFC6241] apply to the new extended RPC operations defined in this document.

10. Contributors

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

Jan Lindblad
Cisco Systems

Email: jlindbla@cisco.com

Chongfeng Xie
China Telecom
Beijing
China

Email: xiechf@chinatelecom.cn

Jason Sterne
Nokia

Email: jason.sterne@nokia.com

Acknowledgements

The authors would like to thank for following for discussions and providing input to this document (ordered by first name): Alex Clemm, Andy Bierman, Balazs Lengyel, Juergen Schoenwaelder, Martin Bjorklund, Mohamed Boucadair, Robert Wilton and Timothy Carey.

References

Normative References

[RFC2119] Bradner, S. and RFC Publisher, "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>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., Bierman, A., Ed., and RFC Publisher, "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6470] Bierman, A. and RFC Publisher, "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, DOI 10.17487/RFC6470, February 2012, <https://www.rfc-editor.org/info/rfc6470>.

[RFC7950] Bjorklund, M., Ed. and RFC Publisher, "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., Watsen, K., and RFC Publisher, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., Wilton, R., and RFC Publisher, "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., Wilton, R., and RFC Publisher, "NETCONF Extensions to Support the Network Management Datastore Architecture", RFC 8526, DOI 10.17487/RFC8526, March 2019, <https://www.rfc-editor.org/info/rfc8526>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., Tripathy, A., and RFC Publisher, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC8641] Clemm, A., Voit, E., and RFC Publisher, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

Informative References

[I-D.ma-netmod-immutable-flag] Ma, Q., Wu, Q., Lengyel, B., and H. Li, "YANG Extension and Metadata Annotation for Immutable Flag", Work in Progress, Internet-Draft, draft-ma-netmod-immutable-flag-04, 20 October 2022, <https://datatracker.ietf.org/doc/html/draft-ma-netmod-immutable-flag-04>.

[RFC7317] Bierman, A., Bjorklund, M., and RFC Publisher, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[RFC8174] Leiba, B. and RFC Publisher, "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. and RFC Publisher, "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., Wilton, R., and RFC Publisher, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

[RFC8808] Wu, Q., Lengyel, B., Niu, Y., and RFC Publisher, "A YANG Data Model for Factory Default Settings", RFC 8808, DOI 10.17487/RFC8808, August 2020, <https://www.rfc-editor.org/info/rfc8808>.

Appendix A. Key Use Cases

Following provides three use cases related to system-defined configuration lifecycle management. The simple interface data model defined in Appendix C.3 of [RFC8342] is used. For each use case, snippets of <running>, <system>, <intended> and <operational> are shown.

A.1. Device Powers On

<running>:

No configuration for "lo0" appears in <running>;

<system>:

```
<interfaces>
  <interface>
    <name>lo0</name>
    <ip-address>127.0.0.1</ip-address>
    <ip-address>::1</ip-address>
  </interface>
</interfaces>
```

<intended>:

```
<interfaces>
  <interface>
    <name>lo0</name>
    <ip-address>127.0.0.1</ip-address>
    <ip-address>::1</ip-address>
  </interface>
</interfaces>
```

<operational>:

```
<interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
            or:origin="or:system">
  <interface>
    <name>lo0</name>
    <ip-address>127.0.0.1</ip-address>
    <ip-address>::1</ip-address>
  </interface>
</interfaces>
```

A.2. Client Commits Configuration

If a client creates an interface "et-0/0/0" but the interface does not physically exist at this point:

```
   <running>:

        <interfaces>
          <interface>
            <name>et-0/0/0</name>
            <description>Test interface</description>
          </interface>
        </interfaces>

   <system>:

        <interfaces>
          <interface>
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
        </interfaces>

   <intended>:

        <interfaces>
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
          <interface>
            <name>et-0/0/0</name>
            <description>Test interface</description>
          </interface>
          <interface>
        </interfaces>

   <operational>:

        <interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                    or:origin="or:intended">
          <interface or:origin="or:system">
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
        </interfaces>
```

A.3. Operator Installs Card into a Chassis

```
   <running>:

        <interfaces>
          <interface>
            <name>et-0/0/0</name>
            <description>Test interface</description>
          </interface>
        </interfaces>

   <system>:

        <interfaces>
          <interface>
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
          <interface>
            <name>et-0/0/0</name>
            <mtu>1500</mtu>
          </interface>
        </interfaces>

   <intended>:

        <interfaces>
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
          <interface>
            <name>et-0/0/0</name>
            <description>Test interface</description>
            <mtu>1500</mtu>
          </interface>
          <interface>
        </interfaces>

   <operational>:
        <interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                    or:origin="or:intended">
          <interface or:origin="or:system">
            <name or:origin>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
         <interface>
            <name>et-0/0/0</name>
            <description>Test interface</description>
            <mtu or:origin="or:system">1500</mtu>
          </interface>
          <interface>
        </interfaces>
```

Appendix B. Changes between Revisions

v00 - v01

* Clarify why client's explicit copy is not preferred but cannot be avoided if resolve-system parameter is not defined

* Clarify active system configuration

* Update the timing when the server's auto copy should be enforced if a resolve-system parameter is used

* Editorial changes

Appendix C. Open Issues tracking

* Should the "with-origin" parameter be supported for <intended>?

Authors' Addresses

Qiufang Ma (editor)
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: maqiufang1@huawei.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: bill.wu@huawei.com

Feng Chong
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: frank.fengchong@huawei.com

Network Working Group
Internet-Draft
Updates: 6020, 7950, 8407, 8525 (if approved)
Intended status: Standards Track
Expires: 16 July 2023

R. Wilton, Ed.
Cisco Systems, Inc.
R. Rahman, Ed.
Graphiant
B. Lengyel, Ed.
Ericsson
J. Clarke
Cisco Systems, Inc.
J. Sterne
Nokia
12 January 2023

Updated YANG Module Revision Handling
draft-ietf-netmod-yang-module-versioning-08

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 a minimum revision suggestion to help document inter-module dependencies. It provides guidelines for managing the lifecycle of YANG modules and individual schema nodes. It provides a mechanism, via the revision label YANG extension, to specify a revision identifier for YANG modules and submodules. This document updates RFC 7950, RFC 6020, RFC 8407 and RFC 8525.

Status of This Memo

Table of Contents

1. Introduction

The current YANG [RFC7950] module update rules require that updates of YANG modules preserve strict backwards compatibility. This has caused problems as described in [I-D.ietf-netmod-yang-versioning-reqs]. This document recognizes the need to sometimes allow YANG modules to evolve with non-backwards-compatible changes, which can 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 when these changes occur, and to manage their effect on clients and the broader YANG ecosystem.

This document defines a flexible versioning solution. Several other documents build on this solution with additional capabilities. [I-D.ietf-netmod-yang-schema-comparison] specifies an algorithm that can be used to compare two revisions of a YANG schema and provide granular information to allow module users to determine if they are impacted by changes between the revisions. The [I-D.ietf-netmod-yang-semver] document extends the module versioning work by introducing a revision label scheme based on semantic versioning. YANG packages [I-D.ietf-netmod-yang-packages] provides a mechanism to group sets of related YANG modules together in order to manage schema and conformance of YANG modules as a cohesive set instead of individually. Finally, [I-D.ietf-netmod-yang-ver-selection] provides a schema selection mechanism that allows a client to choose which schemas to use when interacting with a server from the available schema that are supported and advertised by the server. These other documents are mentioned here as informative references. Support of the other documents is not required in an implementation in order to take advantage of the mechanisms and functionality offered by this module versioning document.

The document 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.

* Updated guidance for revision selection on imports and a YANG extension statement allowing YANG module imports to document 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 and submodule descriptions, to report how "deprecated" and "obsolete" nodes are handled by a server, and to clarify how module imports are resolved when multiple revisions 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.

Note to RFC Editor (To be removed by RFC Editor)

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 and [RFC6020] section 10. Section 3 describes modifications to YANG revision handling and update rules, and Section 4.1 describes a YANG extension statement to describe potential YANG import revision dependencies.

This document updates [RFC7950] section 5.2, [RFC6020] section 5.2 and [RFC8407] section 3.2. Section 3.4.1 describes the use of a revision label in the name of a file containing a YANG module or submodule.

This document updates [RFC7950] section 5.6.5 and [RFC8525]. Section 5.1 defines how a client of a YANG library datastore schema resolves ambiguous imports for modules which are not "import-only".

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.

This document updates [RFC8525] with augmentations to include revision labels in the YANG library data and two boolean leafs to indicate whether status deprecated and status obsolete schema nodes are implemented by the server.

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 makes use of the following terminology introduced in the YANG 1.1 Data Modeling Language [RFC7950]:

* schema node

In addition, this document uses the following 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] and [RFC6020] assume, but do not explicitly state, that the revision history for a YANG module or submodule is strictly linear, i.e., it is prohibited to have two independent revisions of a YANG module or submodule that are both directly derived from the same parent revision.

This document clarifies [RFC7950] and [RFC6020] to explicitly allow non-linear development of YANG module and submodule revisions, so that they MAY have multiple revisions that directly derive from the same parent revision. As per [RFC7950] and [RFC6020], YANG module and submodule revisions continue to be uniquely identified by their revision date, and hence all revisions of a given module or submodule MUST have unique revision dates.

For a given YANG module revision, revision B is defined as being derived from revision A, if revision A is listed in the revision history of revision B. Although this document allows for a branched revision history, a given YANG module revision history does not contain all revisions in all possible branches, it only lists those from which is was derived, i.e., the module revision’s history describes a single path of derived revisions back to the root of the module’s revision history.

A corollary to the text above is that the ancestry (derived relationship) between two module or submodule revisions cannot be determined by comparing the module or submodule revision date or label alone - the revision history must be consulted.

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 to ensure that the module’s content is uniquely defined, the module’s "include" statements SHOULD use "revision-date" substatements to specify the exact revision date of each included submodule. When a module does not include its submodules by revision-date, the revision of submodules used cannot be derived from the including module. Mechanisms such as YANG packages [I-D.ietf-netmod-yang-packages], and YANG library [RFC8525], MAY be used to specify the exact submodule revisions used when the submodule revision date is not constrained by the "include" statement.

[RFC7950] section 11 and [RFC6020] section 10 require that all updates to a YANG module are BC to the previous revision of the module. This document introduces a method to indicate that an NBC change has occurred between module revisions: this is done by using a new "non-backwards-compatible" YANG extension statement in the module revision history.

Two revisions of a module or submodule MAY have identical content except for the revision history. This could occur, for example, if a module or submodule has a branched history and identical changes are applied in multiple branches.

3.1. Updating a YANG module with a new revision

This section updates [RFC7950] section 11 and [RFC6020] section 10 to refine the rules for permissible changes when a new YANG module revision is created.

A new module revision MAY contain NBC changes, e.g., the semantics of an existing data-node definition MAY be changed in an NBC manner without requiring a new data-node definition with a new identifier. A YANG extension, defined in Section 3.2, is used to signal the potential for incompatibility to existing module users and readers.

Note that NBC changes often create problems for clients, thus it is recommended to avoid making them.

As per [RFC7950] and [RFC6020], all published revisions of a module are given a new unique revision date. This applies even for module revisions containing (in the module or included submodules) only changes to any whitespace, formatting, comments or line endings (e.g., DOS vs UNIX).

3.1.1. Backwards-compatible rules

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 and [RFC6020] section 10, 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 a non-backwards-compatible change.

* YANG schema nodes with a "status" "obsolete" substatement MAY be removed from published modules, and the removal is classified as a backwards-compatible change. In some circumstances it may be helpful to retain the obsolete definitions since their identifiers may still be referenced by other modules and to ensure that their identifiers are not reused with a different meaning.

* A statement that is defined using the YANG "extension" statement MAY be added, removed, or changed, if it does not change the semantics of the module. Extension statement definitions SHOULD specify whether adding, removing, or changing statements defined by that extension are backwards-compatible or non-backwards-compatible.

* Any change made to the "revision-date" or "recommended-min" substatements of an "import" statement, including adding new "revision-date" or "recommended-min" substatements, changing the argument of any "revision-date" or "recommended-min" substatetements, or removing any "revision-date" or "recommended-min" substatements, is classified as backwards-compatible.

* Any changes (including whitespace or formatting changes) that do not change the semantic meaning of the module are backwards-compatible.

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. non-backwards-compatible extension statement

The "rev:non-backwards-compatible" 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:non-backwards-compatible" extension statement MUST be added as a substatement to the "revision" statement.

Adding, modifying or removing a "rev:non-backwards-compatible" extension statement is considered to be a BC change.

3.3. Removing revisions from the revision history

Authors may wish to remove revision statements from a module or submodule. Removal of revision information may be desirable for a number of reasons including reducing the size of a large revision history, or removing a revision that should no longer be used or imported. Removing revision statements is allowed, but can cause issues and SHOULD NOT be done without careful analysis of the potential impact to users of the module or submodule. Doing so can lead to import breakages when import by recommended-min is used. Moreover, truncating history may cause loss of visibility of when non-backwards-compatible changes were introduced.

An author MAY remove a contiguous sequence of entries from the end (i.e., oldest entries) of the revision history. This is acceptable even if the first remaining (oldest) revision entry in the revision history contains a rev:non-backwards-compatible substatement.

An author MAY remove a contiguous sequence of entries in the revision history as long as the presence or absence of any existing rev:non-backwards-compatible substatements on all remaining entries still accurately reflect the compatibility relationship to their preceding entries remaining in the revision history.

The author MUST NOT remove the first (i.e., newest) revision entry in the revision history.

Example revision history:

```
revision 2020-11-11 {
  rev:label 4.0.0;
  rev:non-backwards-compatible;
}

revision 2020-08-09 {
  rev:label 3.0.0;
  rev:non-backwards-compatible;
}

revision 2020-06-07 {
  rev:label 2.1.0;
}

revision 2020-02-10 {
  rev:label 2.0.0;
  rev:non-backwards-compatible;
}

revision 2019-10-21 {
  rev:label 1.1.3;
}

revision 2019-03-04 {
  rev:label 1.1.2;
}

revision 2019-01-02 {
  rev:label 1.1.1;
}
```

In the revision history example above, removing the revision history entry for 2020-02-10 would also remove the rev:non-backwards-compatible annotation and hence the resulting revision history would incorrectly indicate that revision 2020-06-07 is backwards-compatible with revisions 2019-01-02 through 2019-10-21 when it is not, and so this change cannot be made. Conversely, removing one or more revisions out of 2019-03-04, 2019-10-21 and 2020-08-09 from the revision history would still retain a consistent revision history, and is acceptable, subject to an awareness of the concerns raised in the first paragraph of this section.

3.4. 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.

A revision label scheme is a set of rules describing how a particular type of revision label operates for versioning YANG modules and submodules. For example, YANG Semver [I-D.ietf-netmod-yang-semver] defines a revision label scheme based on Semver 2.0.0 [semver]. Other documents may define other YANG revision label schemes.

Submodules MAY use a revision label scheme. When they use a revision label scheme, submodules MAY use a revision label scheme that is different from the one used in the including module.

The revision label space of submodules is separate from the revision label space of the including module. A change in one submodule MUST result in a new revision label of that submodule and the including module, but the actual values of the revision labels in the module and submodule could be completely different. A change in one submodule does not result in a new revision label in another submodule. A change in a module revision label does not necessarily mean a change to the revision label in all included submodules.

If a revision has an associated revision label, then it may be used instead of the revision date in a "rev:recommended-min" extension statement argument.

A specific revision label identifies a specific revision of the module. If two YANG modules contain the same module name and the same revision label (and hence also the same revision-date) in their latest revision statement, then the file contents of the two modules, including the revision history, MUST be identical.

3.4.1. File names

This section updates [RFC7950] section 5.2, [RFC6020] section 5.2 and [RFC8407] section 3.2

If a revision has an associated revision label, then it is RECOMMENDED that the name of the file for that revision be of the form:

```
module-or-submodule-name ['#' revision-label] ( '.yang' / '.yin' )

  E.g., acme-router-module#2.0.3.yang
```

YANG module (or submodule) files may be identified using either the revision-date (as per [RFC8407] section 3.2) or the revision label.

3.4.2. Revision label scheme extension statement

The optional "rev:revision-label-scheme" extension statement is used to indicate which revision label scheme a module or submodule uses. There MUST NOT be more than one revision label scheme in a module or submodule. The mandatory argument to this extension statement:

* specifies the revision label scheme used by the module or submodule

* is defined in the document which specifies the revision label scheme

* MUST be an identity derived from "revision-label-scheme-base".

The revision label scheme used by a module or submodule SHOULD NOT change during the lifetime of the module or submodule. If the revision label scheme used by a module or submodule is changed to a new scheme, then all revision label statements that do not conform to the new scheme MUST be replaced or removed.

3.5. Examples for updating the YANG module revision history

The following diagram, explanation, and module history illustrates how the branched revision history, "non-backwards-compatible" extension statement, and revision "label" extension statement could be used:

Example YANG module with branched revision history.

```
          Module revision date        Revision label
            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
                |
            2019-06-01                 <- 3.1.0
```

The tree diagram above illustrates how an example module's revision history might evolve, over time. For example, the tree might represent the following changes, listed in chronological order from the oldest revision to the newest revision:

Example module, revision 2019-06-01:

```
   module example-module {

     namespace "urn:example:module";
     prefix "prefix-name";
     rev:revision-label-scheme "yangver:yang-semver";

     import ietf-yang-revisions { prefix "rev"; }
     import ietf-yang-semver { prefix "yangver"; }

     description
       "to be completed";

     revision 2019-06-01 {
       rev:label 3.1.0;
       description "Add new functionality.";
     }

     revision 2019-03-01 {
       rev:label 3.0.0;
       rev:non-backwards-compatible;
       description
         "Add new functionality. Remove some deprecated nodes.";
     }

     revision 2019-02-01 {
       rev:label 2.0.0;
       rev:non-backwards-compatible;
       description "Apply bugfix to pattern statement";
     }

     revision 2019-01-01 {
       rev:label 1.0.0;
       description "Initial revision";
     }

     //YANG module definition starts here
   }
```

Example module, revision 2019-05-01:

```
   module example-module {

     namespace "urn:example:module";
     prefix "prefix-name";
     rev:revision-label-scheme "yangver:yang-semver";

     import ietf-yang-revisions { prefix "rev"; }
     import ietf-yang-semver { prefix "yangver"; }

     description
       "to be completed";

     revision 2019-05-01 {
       rev:label 2.2.0;
       description "Backwards-compatible bugfix to enhancement.";
     }

     revision 2019-04-01 {
       rev:label 2.1.0;
       description "Apply enhancement to older release train.";
     }

     revision 2019-02-01 {
       rev:label 2.0.0;
       rev:non-backwards-compatible;
       description "Apply bugfix to pattern statement";
     }

     revision 2019-01-01 {
       rev:label 1.0.0;
       description "Initial revision";
     }

     //YANG module definition starts here
   }
```

4. Guidance for revision selection on imports

[RFC7950] and [RFC6020] allow YANG module "import" statements to optionally require the imported module to have a specific revision date. In practice, importing a module with an exact revision date can be too restrictive because it requires the importing module to be updated whenever any change to the imported module occurs, and hence section Section 7.1 suggests that authors do not restrict YANG module imports to exact revision dates.

Instead, for conformance purposes (section 5.6 of [RFC7950]), the recommended approach for defining the relationship between specific YANG module revisions is to specify the relationships outside of the YANG modules, e.g., via YANG library [RFC8525], YANG packages [I-D.ietf-netmod-yang-packages], a filesystem directory containing a set of consistent YANG module revisions, or a revision control system commit label.

4.1. Recommending a minimum revision for module imports

Although the previous section indicates that the actual relationship constraints between different revisions of YANG modules should be specified outside of the modules, in some scenarios YANG modules are designed to be loosely coupled, and implementors may wish to select sets of YANG module revisions that are expected to work together. For these cases it can be helpful for a module author to provide guidance on a recommended minimum revision that is expected to satisfy an YANG import. E.g., the module author may know of a dependency on a type or grouping that has been introduced in a particular imported YANG module revision. Although there can be no guarantee that all derived future revisions from the particular imported module will necessarily also be compatible, older revisions of the particular imported module are very unlikely to ever be compatible.

This document introduces a new YANG extension statement to provide guidance to module implementors on a recommended minimum module revision of an imported module that is anticipated to be compatible. This statement has been designed to be machine-readable so that tools can parse the minimum revision extension statement and generate warnings if appropriate, but this extension statement does not alter YANG module conformance of valid YANG module versions in any way, and specifically it does not alter the behavior of the YANG module import statement from that specified in [RFC7950].

The ietf-revisions module defines the "recommended-min" extension statement, a substatement to the YANG "import" statement, to allow for a "minimum recommended revision" to be documented:

> The argument to the "recommended-min" extension statement is a revision date or a revision label.

A particular revision of an imported module adheres to an import's "recommended-min" extension statement if the imported module's revision history contains a revision statement with a matching revision date or revision label. Removing entries from a module's revision history may cause a particular revision to no longer satisfy an import's "recommended-min" statement if the revision-date or label is no longer present in the module's revision history; further described in Section 3.3 and Section 7.1.

The "recommended-min" extension statement MAY be specified multiple times, allowing a set of recommended minimum revisions to be documented. Module implementors are recommended to pick a module revision that adheres to any of the "recommended-min" statements.

Adding, modifying or removing a "recommended-min" extension statement is a BC change.

4.1.1. 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:

```
       Module revision date        Revision label
         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
             |
         2019-06-01                  <- 3.1.0
```

4.1.1.1. Example 1

This example recommends module revisions for import 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:recommended-min 2019-02-01;
}
```

Alternatively, the first example could have used the revision label "2.0.0" instead, which selects the same set of revisions/labels.

```
import example-module {
  rev:recommended-min 2.0.0;
}
```

4.1.1.2. Example 2

This example recommends module revisions for import 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 number than 2019-04-01/2.1.0 it is not a derived revision, and hence it is not a recommended revision for import.

```
import example-module {
  rev:recommended-min 2.1.0;
}
```

4.1.1.3. Example 3

This example recommends module revisions for import that are 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:recommended-min 2019-04-01;
  rev:recommended-min 2019-06-01;
}
```

5. Updates to ietf-yang-library

This document updates YANG 1.1 [RFC7950] and YANG library [RFC8525] to clarify how ambiguous module imports are resolved. It also defines the YANG module, ietf-yang-library-revisions, that augments YANG library [RFC8525] with revision labels and two leafs to indicate how a server implements deprecated and obsolete schema nodes.

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] section 5.6.5 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-yang-library-revisions" YANG module has the following structure (using the notation defined in [RFC8340]):

```
module: ietf-yang-library-revisions
  augment /yanglib:yang-library/yanglib:module-set/yanglib:module:
    +--ro revision-label?   rev:revision-label
  augment /yanglib:yang-library/yanglib:module-set/yanglib:module
            /yanglib:submodule:
    +--ro revision-label?   rev:revision-label
  augment /yanglib:yang-library/yanglib:module-set
            /yanglib:import-only-module/yanglib:submodule:
    +--ro revision-label?   rev:revision-label
  augment /yanglib:yang-library/yanglib:schema:
    +--ro deprecated-nodes-implemented?   boolean
    +--ro obsolete-nodes-absent?          boolean
```

5.2.1. Advertising revision-label

The ietf-yang-library-revisions YANG module augments the "module" and "submodule" lists in ietf-yang-library with "revision-label" leafs to optionally declare the revision label associated with each module and submodule.

5.2.2. Reporting how deprecated and obsolete nodes are handled

The ietf-yang-library-revisions YANG module augments YANG library with two boolean leafs to allow a server to report how it implements status "deprecated" and status "obsolete" schema nodes. The leafs are:

deprecated-nodes-implemented: If set to "true", this leaf indicates that all schema nodes with a status "deprecated" are implemented equivalently as if they had status "current"; otherwise deviations MUST be used to explicitly remove "deprecated" nodes from the schema. If this leaf is set to "false" or absent, then the behavior is unspecified.

obsolete-nodes-absent: If set to "true", this leaf indicates that the server does not implement any status "obsolete" schema nodes. If this leaf is set to "false" or absent, then the behaviour is unspecified.

Servers SHOULD set both the "deprecated-nodes-implemented" and "obsolete-nodes-absent" leafs to "true".

If a server does not set the "deprecated-nodes-implemented" leaf to "true", then clients MUST NOT rely solely on the "rev:non-backwards-compatible" 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 [RFC9195] 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.ietf-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.ietf-netmod-yang-semver].

NBC changes to YANG modules may cause problems to clients, who are consumers of YANG models, and hence YANG module authors SHOULD 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:non-backwards-compatible" statement MUST be added 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 dependencies 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 to remove, or curtail, the older description statements associated with the previous revisions.

The "rev:recommended-min" extension MAY 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 SHOULD use the "revision-date" statement to include specific submodule revisions. The revision of the including module MUST be updated when any included submodule has changed.

In some cases a module or submodule revision that is not strictly NBC by the definition in Section 3.1.2 of this specification may include the "non-backwards-compatible" statement. Here is an example when adding the statement may be desirable:

* A "config false" leaf had its value space expanded (for example, a range was increased, or additional enum values were added) and the author or server implementor feels there is a significant compatibility impact for clients and users of the module or submodule

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 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". At some point in the future, when support is removed for the data node, there are two options. The first, and preferred, option is to keep the data node definition in the model and change the status to "obsolete". The second option is to simply remove the data node from the model, but this has the risk of breaking modules which import the modified module, and the removed identifier may be accidently reused in a future revision.

2. For deprecated data nodes the "description" statement 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 "description". The reason for deprecating the node can also be included in the "description" if it is deemed to be of potential interest to the user.

3. For obsolete data nodes, it is RECOMMENDED to keep the above information, from when the node had status "deprecated", which is still relevant.

4. When obsoleting or deprecating data nodes, the "deprecated" or "obsolete" status SHOULD be applied at the highest possible level in the data tree. For clarity, the "status" statement SHOULD also be applied to all descendent data nodes, but the additional status related information does not need to be repeated if it does not introduce any additional information.

5. NBC changes which can break imports SHOULD be avoided because of the impact on the importing module. The importing modules could get broken, e.g., if an augmented node in the importing module has been removed from the imported module. Alternatively, the schema of the importing modules could undergo an NBC change due to the NBC change in the imported module, e.g., if a node in a grouping has been removed. As described in Appendix B.1, instead of removing a node, that node SHOULD first be deprecated and then obsoleted.

See Appendix B 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, revision label scheme, and importing by revision.

```
   <CODE BEGINS> file "ietf-yang-revisions@2022-11-29.yang"
   module ietf-yang-revisions {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-yang-revisions";
     prefix rev;

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

        Author:   Joe Clarke
                 <mailto:jclarke@cisco.com>

        Author:   Reshad Rahman
                 <mailto:reshad@yahoo.com>

        Author:   Robert Wilton
                 <mailto:rwilton@cisco.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) 2002 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 Revised 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.

        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 2022-11-29 {
       rev:label "1.0.0-draft-ietf-netmod-yang-module-versioning-08";
       description
         "Initial version.";
       reference
         "XXXX: Updated YANG Module Revision Handling";
     }

     typedef revision-date {
       type string {
         pattern '[0-9]{4}-(1[0-2]|0[1-9])-(0[1-9]|[1-2][0-9]|3[0-1])';
       }
       description
         "A date associated with a YANG revision.

          Matches dates formatted as YYYY-MM-DD.";
       reference
         "RFC 7950: The YANG 1.1 Data Modeling Language";
     }

     typedef revision-label {
       type string {
         length "1..255";
         pattern '[a-zA-Z0-9,\-_.+]+';
         pattern '[0-9]{4}-[0-9]{2}-[0-9]{2}' {
           modifier "invert-match";
           error-message
             "The revision-label must not match a revision-date.";
         }
       }
       description
         "A label associated with a YANG revision.

          Alphanumeric characters, comma, hyphen, underscore, period
          and plus are the only accepted characters. MUST NOT match
          revision-date or pattern similar to a date.";
       reference
         "XXXX: Updated YANG Module Revision Handling;
          Section 3.3, Revision label";
     }

     typedef revision-date-or-label {
       type union {
         type revision-date;
         type revision-label;
       }
       description
         "Represents either a YANG revision date or a revision label";
     }

     extension non-backwards-compatible {
       description
         "This statement is used to indicate YANG module revisions that
          contain non-backwards-compatible changes.

          The statement MUST only be a substatement of the 'revision'
          statement.  Zero or one 'non-backwards-compatible' statements
          per parent statement is allowed.  No substatements for this
          extension have been standardized.

          If a revision of a YANG module contains changes, relative to
          the preceding revision in the revision history, that do not
          conform to the backwards-compatible module update rules
          defined in RFC-XXX, then the 'non-backwards-compatible'
          statement MUST be added as a substatement to the revision
          statement.

          Conversely, if a revision does not contain a
          'non-backwards-compatible' statement then all changes,
          relative to the preceding revision in the revision history,
          MUST be backwards-compatible.

          A new module revision that only contains changes that are
          backwards-compatible SHOULD NOT include the
          'non-backwards-compatible' statement.  An example of when an
          author might add the 'non-backwards-compatible' statement is
          if they believe a change could negatively impact clients even
          though the backwards compatibility rules defined in RFC-XXXX
          classify it as a backwards-compatible change.

          Add, removing, or changing a 'non-backwards-compatible'
          statement is a backwards-compatible version change.";
       reference
         "XXXX: Updated YANG Module Revision Handling;
          Section 3.2,
          non-backwards-compatible revision extension statement";
     }

     extension label {
       argument revision-label;
       description
```

```
         "The revision label can be used to provide an additional
          versioning identifier associated with a module or submodule
          revision.  One such scheme that could be used is [XXXX:
          ietf-netmod-yang-semver].

          The format of the revision label argument MUST conform to the
          pattern defined for the revision label typedef in this module.

          The statement MUST only be a substatement of the revision
          statement.  Zero or one revision label statements per parent
          statement are allowed.  No substatements for this extension
          have been standardized.

          Revision labels MUST be unique amongst all revisions of a
          module or submodule.

          Adding a revision label is a backwards-compatible version
          change.  Changing or removing an existing revision label in
          the revision history is a non-backwards-compatible version
          change, because it could impact any references to that
          revision label.";
       reference
         "XXXX: Updated YANG Module Revision Handling;
          Section 3.3, Revision label";
     }

     extension revision-label-scheme {
       argument revision-label-scheme-base;
       description
         "The revision label scheme specifies which revision label
          scheme the module or submodule uses.

          The mandatory revision-label-scheme-base argument MUST be an
          identity derived from revision-label-scheme-base.

          This extension is only valid as a top-level statement, i.e.,
          given as as a substatement to 'module' or 'submodule'.  No
          substatements for this extension have been standardized.

          This extension MUST be used if there is a revision label
          statement in the module or submodule.

          Adding a revision label scheme is a backwards-compatible
          version change.  Changing a revision label scheme is a
          non-backwards-compatible version change, unless the new
          revision label scheme is backwards-compatible with the
          replaced revision label scheme.  Removing a revision label
          scheme is a non-backwards-compatible version change.";
       reference
         "XXXX: Updated YANG Module Revision Handling;
          Section 3.3.1, Revision label scheme extension statement";
     }

     extension recommended-min {
       argument revision-date-or-label;
       description
         "Recommends the revision of the module that may be imported to
          one that matches or is derived from the specified
          revision-date or revision label.

          The argument value MUST conform to the
          'revision-date-or-label' defined type.

          The statement MUST only be a substatement of the import
          statement.  Zero, one or more 'recommended-min' statements per
          parent statement are allowed.  No substatements for this
          extension have been standardized.

          If specified multiple times, then any module revision that
          satisfies at least one of the 'recommended-min' statements is
          an acceptable recommended revision for import.

          A particular revision of an imported module adheres to an
          import's 'recommended-min' extension statement if the imported
          module's revision history contains a revision statement with a
          matching revision date or revision label.

          Adding, removing or updating a 'recommended-min' statement to
          an import is a backwards-compatible change.";
       reference
         "XXXX: Updated YANG Module Revision Handling; Section 4,
          Recommending a minimum revision for module imports";
     }

     identity revision-label-scheme-base {
       description
         "Base identity from which all revision label schemes are
          derived.";
       reference
         "XXXX: Updated YANG Module Revision Handling;
          Section 3.3.1, Revision label scheme extension statement";
     }
   }
   <CODE ENDS>
```

YANG module with augmentations to YANG Library to revision labels

```
   <CODE BEGINS> file "ietf-yang-library-revisions@2021-11-04.yang"
   module ietf-yang-library-revisions {
     yang-version 1.1;
     namespace
       "urn:ietf:params:xml:ns:yang:ietf-yang-library-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 Library";
     }

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

        Author:   Joe Clarke
                  <mailto:jclarke@cisco.com>

        Author:   Reshad Rahman
                  <mailto:reshad@yahoo.com>

        Author:   Robert Wilton
                  <mailto:rwilton@cisco.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) 2002 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 Revised 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.

        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 label version with 1.0.0 and
     // remove this note.

     revision 2021-11-04 {
       rev:label "1.0.0-draft-ietf-netmod-yang-module-versioning-05";
       description
         "Initial revision";
       reference
         "XXXX: Updated YANG Module Revision Handling";
     }

     // library 1.0 modules-state is not augmented with revision-label

     augment "/yanglib:yang-library/yanglib:module-set/yanglib:module" {
       description
         "Add a revision label to module information";
       leaf revision-label {
         type rev:revision-label;
         description
           "The revision label associated with this module revision.
            The label MUST match the revision 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:module-set/yanglib:module/"
     + "yanglib:submodule" {
       description
         "Add a revision label to submodule information";
       leaf revision-label {
         type rev:revision-label;
         description
           "The revision label associated with this submodule revision.
            The label MUST match the revision label value in the
            specific revision of the submodule included by 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:module-set/"
           + "yanglib:import-only-module" {
       description
         "Add a revision label to module information";
       leaf revision-label {
         type rev:revision-label;
         description
           "The revision label associated with this module revision.
            The label MUST match the revision label value in the
            specific revision of the module included in this
            module-set.";
         reference
           "XXXX: Updated YANG Module Revision Handling;
            Section 5.2.1, Advertising revision-label";
       }
     }

     augment "/yanglib:yang-library/yanglib:module-set/"
           + "yanglib:import-only-module/yanglib:submodule" {
       description
         "Add a revision label to submodule information";
       leaf revision-label {
         type rev:revision-label;
         description
           "The revision label associated with this submodule revision.
            The label MUST match the rev:label value in the specific
            revision of the submodule included by the import-only-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 boolean;
         description
           "If set to true, this leaf indicates that all schema nodes
            with a status 'deprecated' are implemented equivalently as
            if they had status 'current'; otherwise deviations MUST be
            used to explicitly remove deprecated nodes from the schema.
            If this leaf is absent or set to false, 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 boolean;
         description
           "If set to true, this leaf indicates that the server does not
            implement any status 'obsolete' schema nodes.  If this leaf
            is absent or set to false, 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. Security considerations

9.1. Security considerations for module revisions

As discussed in the introduction of this document, YANG modules occasionally undergo changes that are not backwards compatible. This occurs in both standards and vendor YANG modules despite the prohibitions in RFC 7950. RFC 7950 also allows nodes to change to status 'obsolete' which can change behavior and compatibility for a client.

The fact that YANG modules change in a non-backwards-compatible manner may have security implications. Such changes should be carefully considered, including the scenarios described below. The rev:non-backwards-compatible extension statement introduced in this document provides an alert that the module or submodule may contain changes that impact users and need to be examined more closely for both compatibility and potential security implications. Flagging the change reduces the risk of introducing silent exploitable vulnerabilities.

When a module undergoes a non-backwards-compatible change, a server may implement different semantics for a given leaf than a client using an older version of the module is expecting. If the particular leaf controls any security functions of the device, or is related to parts of the configuration or state that are sensitive from a security point of view, then the difference in behavior between the old and new revisions needs to be considered carefully. In particular, changes to the default of the leaf should be examined.

Implementors and users should also consider impact to data node access control rules (e.g. The Network Configuration Access Control Model (NACM) [RFC8341]) in the face of non-backwards-compatible changes. Access rules may need to be adjusted when a new module revision is introduced that contains a non-backwards-compatible change.

If the changes to a module or submodule have security implications, it is recommended to highlight those implications in the description of the revision statement.

9.2. Security considerations for the modules defined in this document

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.

This document does not define any new protocol or data nodes that are writable.

This document updates YANG Library [RFC8525] with augmentations to include revision labels in the YANG library data and two boolean leafs to indicate whether status deprecated and status obsolete schema nodes are implemented by the server. These read-only augmentations do not add any new security considerations beyond those already present in [RFC8525].

10. IANA Considerations

10.1. YANG Module Registrations

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-revisions
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-yang-library-revisions
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.
```

The following YANG module is requested to be registred in the "IANA Module Names" [RFC6020]. Following the format in RFC 6020, the following registrations are requested:

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-yang-library-revisions module:

Name: ietf-yang-library-revisions

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-library-revisions

Prefix: yl-rev

Reference: [RFCXXXX]

10.2. Guidance for versioning in IANA maintained YANG modules

Note for IANA (to be removed by the RFC editor): Please check that the registries and IANA YANG modules are referenced in the appropriate way.

IANA is responsible for maintaining and versioning YANG modules that are derived from other IANA registries. For example, "iana-if-type.yang" [IfTypeYang] is derived from the "Interface Types (ifType) IANA registry" [IfTypesReg], and "iana-routing-types.yang" [RoutingTypesYang] is derived from the "Address Family Numbers" [AddrFamilyReg] and "Subsequent Address Family Identifiers (SAFI) Parameters" [SAFIReg] IANA registries.

Normally, updates to the registries cause any derived YANG modules to be updated in a backwards-compatible way, but there are some cases where the registry updates can cause non-backward-compatible updates to the derived YANG module. An example of such an update is the 2020-12-31 revision of iana-routing-types.yang [RoutingTypesDecRevision], where the enum name for two SAFI values was changed.

In all cases, IANA MUST follow the versioning guidance specified in Section 3.1, and MUST include a "rev:non-backwards-compatible" substatement to the latest revision statement whenever an IANA maintained module is updated in a non-backwards-compatible way, as described in Section 3.2.

Note: For published IANA maintained YANG modules that contain non-backwards-compatible changes between revisions, a new revision should be published with the "rev:non-backwards-compatible" substatement retrospectively added to any revisions containing non-backwards-compatible changes.

Non-normative examples of updates to enumeration types in IANA maintained modules that would be classified as non-backwards-compatible changes are: Changing the status of an enumeration typedef to obsolete, changing the status of an enum entry to obsolete, removing an enum entry, changing the identifier of an enum entry, or changing the described meaning of an enum entry.

Non-normative examples of updates to enumeration types in IANA maintained modules that would be classified as backwards-compatible changes are: Adding a new enum entry to the end of the enumeration, changing the status or an enum entry to deprecated, or improving the description of an enumeration that does not change its defined meaning.

Non-normative examples of updates to identity types in IANA maintained modules that would be classified as non-backwards-compatible changes are: Changing the status of an identity to obsolete, removing an identity, renaming an identity, or changing the described meaning of an identity.

Non-normative examples of updates to identity types in IANA maintained modules that would be classified as backwards-compatible changes are: Adding a new identity, changing the status or an identity to deprecated, or improving the description of an identity that does not change its defined meaning.

11. References

11.1. Normative References

[I-D.ietf-netmod-yang-semver]
Clarke, J., Wilton, R., Rahman, R., Lengyel, B., Sterne, J., and B. Claise, "YANG Semantic Versioning", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-semver-08, 24 October 2022, <https://www.ietf.org/archive/id/draft-ietf-netmod-yang-semver-08.txt>.

[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>.

[RFC6020] Bjorklund, M., Ed. and RFC Publisher, "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., Bierman, A., Ed., and RFC Publisher, "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. and RFC Publisher, "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>.

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

[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>.

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

[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

[AddrFamilyReg]
"Address Family Numbers IANA Registry", <https://www.iana.org/assignments/address-family-numbers/address-family-numbers.xhtml>.

[I-D.clacla-netmod-yang-model-update]
Claise, B., Clarke, J., Lengyel, B., and K. D'Souza, "New YANG Module Update Procedure", Work in Progress, Internet-Draft, draft-clacla-netmod-yang-model-update-06, 2 July 2018, <https://www.ietf.org/archive/id/draft-clacla-netmod-yang-model-update-06.txt>.

[I-D.ietf-netmod-yang-packages]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and B. Wu, "YANG Packages", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-packages-03, 4 March 2022, <https://www.ietf.org/archive/id/draft-ietf-netmod-yang-packages-03.txt>.

[I-D.ietf-netmod-yang-schema-comparison]
Wilton, R., "YANG Schema Comparison", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-schema-comparison-01, 2 November 2020, <https://www.ietf.org/archive/id/draft-ietf-netmod-yang-schema-comparison-01.txt>.

[I-D.ietf-netmod-yang-ver-selection]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and B. Wu, "YANG Schema Selection", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-ver-selection-00, 17 March 2020, <https://www.ietf.org/archive/id/draft-ietf-netmod-yang-ver-selection-00.txt>.

[I-D.ietf-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-versioning-reqs-07, 10 July 2022, <https://www.ietf.org/archive/id/draft-ietf-netmod-yang-versioning-reqs-07.txt>.

[IfTypesReg]
"Interface Types (ifType) IANA Registry", <https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-5>.

[IfTypeYang]
"iana-if-type YANG Module", <https://www.iana.org/assignments/iana-if-type/iana-if-type.xhtml>.

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

[RFC9195] Lengyel, B. and B. Claise, "A File Format for YANG Instance Data", RFC 9195, DOI 10.17487/RFC9195, February 2022, <https://www.rfc-editor.org/info/rfc9195>.

[RoutingTypesDecRevision]
"2020-12-31 revision of iana-routing-types.yang", <https://www.iana.org/assignments/yang-parameters/iana-routing-types@2020-12-31.yang>.

[RoutingTypesYang]
"iana-routing-types YANG Module", <https://www.iana.org/assignments/iana-routing-types/iana-routing-types.xhtml>.

[SAFIReg] "Subsequent Address Family Identifiers (SAFI) Parameters IANA Registry", <https://www.iana.org/assignments/safi-namespace/safi-namespace.xhtml>.

[semver] "Semantic Versioning 2.0.0", <https://www.semver.org>.

Appendix A. Examples of changes that are NBC

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 remove any previously allowed values from 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.

Appendix B. Examples of applying the NBC change guidelines

The following sections give steps that could be taken for making NBC changes to a YANG module or submodule using the incremental approach described in section Section 7.1.1.

The examples are all for "config true" nodes.

B.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 schema node's status is changed to "deprecated" and the node is supported for some period of time (e.g. one year). This is a BC change.

2. When the schema node is not supported anymore, its status is changed to "obsolete" and the "description" updated. This is an NBC change.

B.2. Changing the type of a leaf node

Changing the type of a leaf node. e.g., a "vpn-id" node of type integer being changed to a string:

1. The status of schema node "vpn-id" is changed to "deprecated" and the node is supported for some period of time (e.g. one year). This is a BC change. The description is updated to indicate that "vpn-name" is replacing this node.

2. A new schema 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 when both schema nodes are supported, the interactions between the two nodes is outside the scope of this document and will vary on a case by case basis. One possible option is to have the server 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 added to it to achieve this. The old node, however, must not have a "when" statement added, or an existing "when" modified to be more restrictive, since this would be an NBC change. In any case, the server could reject the old node from being set if the new node is already set.

4. When the schema node "vpn-id" is not supported anymore, its status is changed to "obsolete" and the "description" is updated. This is an NBC change.

B.3. Reducing the range of a leaf node

Reducing the range of values of a leaf-node, e.g., consider a "vpn-id" schema node of type uint32 being changed from range 1..5000 to range 1..2000:

1. If all values which are being removed were never supported, e.g., if a vpn-id of 2001 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 3333 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 B.2

B.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 is supported for some period of time (e.g. one year). This is a BC change. The description is updated to indicate the new list that is replacing this list.

2. A new list is created in the same location with the same descendant schema nodes but with "dest-address" as key. Finding an appropriate name for the new list can be difficult. 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 when both lists are supported, the interactions between the two lists is outside the scope of this document and will vary on a case by case basis. One possible option is to have the server prevent entries in the new list from being created if the old list already has entries (and vice-versa).

4. When list "sessions" is not available anymore, its status is changed to "obsolete" and the "description" is updated. This is an NBC change.

B.5. Renaming a node

A leaf or container schema node 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 is supported for some period of time (e.g. one year). This is a BC change. The description is updated to indicate the node that is replacing this node.

2. The new schema 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 when both nodes are available, the interactions between the two nodes is outside the scope of this document and will vary on a case by case basis. One possible option is to have the server 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 added to it to achieve this. The old node, however, must not have a "when" statement added, or an existing "when" modified to be more restrictive, since this would be an NBC change. In any case, the server could reject the old node from being set if the new node is already set.

4. When node "ip-adress" is not available anymore, its status is changed to "obsolete" and the "description" is updated. This is an NBC change.

Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The authors and the following individuals are (or have been) members of the design team and have worked on the YANG versioning project:

Benoit Claise
benoit.claise@huawei.com

Bo Wu
lana.wubo@huawei.com

Ebben Aries
exa@juniper.net

Jan Lindblad
lindbla@cisco.com

Juergen Schoenwaelder
j.shoenwaelder@jacobs-university.de

Mahesh Jethanandani
mjethanandani@gmail.com

Michael (Wangzitao)
wangzitao@huawei.com

Per Andersson
perander@cisco.com

Qin Wu
bill.wu@huawei.com

The initial revision of this document was refactored and built upon [I-D.clacla-netmod-yang-model-update]. We would like to thank Kevin D’Souza and Benoit Claise for their initial work in this problem space.

Discussions on the use of Semver for YANG versioning has been held with authors of the OpenConfig YANG models. We would like to thank both Anees Shaikh and Rob Shakir for their input into this problem space.

We would also like to thank Lou Berger, Andy Bierman, Martin Bjorklund, Italo Busi, Tom Hill, Scott Mansfield, and Kent Watsen for their contributions and review comments.

Authors’ Addresses

Robert Wilton (editor)
Cisco Systems, Inc.
Email: rwilton@cisco.com

Reshad Rahman (editor)
Graphiant
Email: reshad@yahoo.com

Balazs Lengyel (editor)
Ericsson
Email: balazs.lengyel@ericsson.com

Joe Clarke
Cisco Systems, Inc.
Email: jclarke@cisco.com

Jason Sterne
Nokia
Email: jason.sterne@nokia.com

Network Working Group   P. Andersson, Ed.
Internet-Draft   R. Wilton
Updates: 7950 (if approved)   Cisco Systems, Inc.
Intended status: Standards Track   11 March 2023
Expires: 12 September 2023

YANG Schema Comparison
draft-ietf-netmod-yang-schema-comparison-02

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

Copyright Notice

Table of Contents

1. Key Issues

{ This section is only to present the current ongoing work, not part of the final draft. }

The contributors have identified several key issues that need attention. This section presents selected key issues which have been discussed together with suggestions for proposed solution or requirements.

1.1. On-wire vs Schema analysis

Should one algorithm be used or two? The consesus reached was to define two separate algorithms, one for on-wire format and one for schema.

On the wire: the focus is on what types of changes affect the client requests and server responses for YANG driven protocols, e.g. NETCONF, RESTCONF, gNMI. If the same requests and responses occur, then there is no "on the wire" impact of the change. For example, changing the name of a "choice" has no impact "on the wire". For many clients, this level of compatiblity is enough.

Schema: any changes that affect the YANG schema in an NBC manner according to the full rules of [I-D.ietf-netmod-yang-module-versioning]. This may be important for clients that, for example, automatically generate code using the YANG and where the change of a typedef name or a choice name could be significant. Also important for other modules that may augment or deviate the schema being compared.

Changes to the module that aren't semantic should raise that there has been editorial changes

Ordering in the schema, RFC 7950 doesn't allow reordering; thus an NBC change.

Open Questions:

Groupings / uses

typedefs, namespaces, choice names, prefixes, module metadata.

* typedef renaming (on-wire, same base type etc)

* Should all editorial (text) diffs be reported?

* What about editorial changes that might change semantics, e.g. a description of a leaf?

* Metadata arguments which relies on the formatted input text. E.g description, contact (etc), extension (how does the user want to tune verbosity level for editorial changes: whitespace, spelling, editorial, potentially-nbc?

* XPath, must, when: don’t normalize XPath expressions

* presence statements

1.2. error-tags, error messages, and other error statements

Error tags and messages might be relied on verbatim by users.

* error-tag: standardized in [RFC6241]

* error-app-tag: arbitrary text ([RFC6241] but also model)

* error-message: arbitrary

Failed must statement, error-message, assumed NBC

Default behaviour is changes to error tags, messages etc are NBC.

1.3. Comparison on module or full schema (YANG artifact, arbitrary blob. Questions

* features

* packages vs directories vs libraries vs artifact

* package specific comparison, package metadata or only looking at the modules

* import only or implemented module

Filter out comparison for a specific subrtree, path etc. Use case for on-wire e.g. yang subscriptions, did the model change fro what is subscribed on?

2. Open Issues

{ This section is only to present the current ongoing work, not part of the final draft. }

The following issues have not ben discussed in any wider extent yet.

2.1. Override/per-node tags

2.2. Separate rules for config vs state

2.3. Tool/report verbosity

* where to report changes (module, grouping, typedef, uses)

* output level (conceptual level or exact strings)

* granularity: error/warning/info level per reported change category

2.4. sub-modules

2.5. Write algorithm in pseudo code or just describe the rules/goals in text?

2.6. Categories in the report: bc, nbc, potentially-nbc, editorial. Allow filtering in the draft without defining it?

One option can be to have a tool option that presents the reason behind the decision, e.g. --details could be used to explain to the user why a certain change was marked as nbc.

Another option is to present reasoning and analysis in deeper levels of verbosity; e.g. one extra level of verbosity, -v, could present the reason for categorizing a change nbc, and an additional extra level of verbosity, e.g. -vv, could also present the detailed analysis the tool made to categorize the change.

2.7. Only for YANG 1.1?

2.8. renamed-from

3. Tool options

{ This section is only to present the current ongoing work, not part of the final draft. }

During the work a list of useful tool options are identified for later discussion and publication in an appendix.

* An option for how to interpret description changes (for the on-wire algorithm) by default, e.g. treat them as editorial or nbc.

* Option: --skip-error-tags, etc

4. 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.ietf-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.ietf-netmod-yang-solutions].

YANG module ’revision-labels’ [I-D.ietf-netmod-yang-module-versioning] and the use of YANG semantic version numbers [I-D.ietf-netmod-yang-semver] can be used to help manage and report changes between revisions of individual YANG modules.

YANG packages [I-D.ietf-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.ietf-netmod-yang-module-versioning] and [I-D.ietf-netmod-yang-packages] define how to classify changes between two module or package revisions, respectively, as backwards compatible or non-backwards-compatible. [I-D.ietf-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'.

5. 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 makes use of the following terminology introduced in the YANG 1.1 Data Modeling Language [RFC7950]:

* schema node

This document uses terminology introduced in the YANG versioning requirements document [I-D.ietf-netmod-yang-versioning-reqs].

This document makes of the following terminology introduced in the YANG Packages [I-D.ietf-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.

* Node compatibility statement: An extension statements (e.g. nbc-change-at) that can be used to indicate the backwards compatibility of individual schema nodes and specific YANG statements.

6. 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.ietf-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.ietf-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.

6.1. YANG module revision scope extension annotations

6.2. Node compatibility extension statements

In addition to the revision extension statement in [I-D.ietf-netmod-yang-module-versioning], this document defines YANG extension statements to indicate compatibility information for individual schema nodes and certain YANG statements.

The node compatibility extension statements are applicable to schema nodes (e.g. leaf, rpc, choice) as defined in [RFC7950], as well as a set of YANG statements (e.g. typedef) as listed in the YANG definition of the nbc-change-at extension in the ietf-yang-revisions module in this document.

While the top level non-backwards-compatible-revision statement is mandatory when there is a non-backwards-compatible change, the node compatibility statements are optional.

For many YANG statements, programmatic tooling can determine whether the changes to a statement between two module revisions constitutes a backwards-compatible or non-backwards-compatible change. However, for some statements, it may be impractical for 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 schema 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.

Three extensions are defined for schema node compatibility information:

nbc-change-at: Indicates a specific YANG statement had a non-backwards-compatible change at a particular module or sub-module revision

bc-change-at: Indicates a specific YANG statement had a backwards-compatible change at a particular module or sub-module revision

editorial-change-at: Indicates a specific YANG statement had an editorial change at a particular module or sub-module revision. The meaning of an editorial change is as per YANG Semver [I-D.ietf-netmod-yang-semver]

When a node compatibility statement is added to a schema node in a sub-module, the revision indicated for the compatibility statement is that of the sub-module.

Adding, modifying or removing any of the node compatibility statements is considered to be a BC change.

The following example illustrates the node compatibility statements:

```
                 container some-stuff {
                   leaf used-to-be-a-string {
                     rev:nbc-change-at "3.0.0" {
                       description "Changed from a string to a uint32.";
                     }
                     type uint32;
                   }
                   leaf fixed-my-description-typo {
                     rev:editorial-change-at "2022-06-03";
                     type string;
                     description "This description used to have a typo."
                   }
                   list sir-changed-a-lot {
                     rev:editorial-change-at "3.0.0";
                     rev:bc-change-at "2.3.0";
                     rev:bc-change-at "1.2.1_non_compatible";
                     description "a list of stuff";
                     ordered-by user;
                     key "foo";
                     leaf foo {
                       type string;
                     }
                     leaf thing {
                       type uint8;
                     }
                   }
```

Note that an individual YANG statement may have a backwards-compatible change in a revision that is non-backwards-compatible (e.g. some other node changed in a non-backwards-compatible fashion in that particular revision).

If changes are ported from one branch of YANG model revisions to another branch, care must be taken with any node compatibilty statements. A simple copy-n-paste should not be used. The node compatibilty statements may incorrectly reference a revision that is not in the history of the new revision. Further, the statements might not apply depending on what the history is like in that new branch (e.g., an NBC change that is ported might not be an NBC change in the new branch). Node compatiblity statements should not be copied over to the new branch. Instead, the changes should be considered as completely new on the new branch, and any compatibility information should be generated from scratch.

When a node compatibility statement is present, that compatibilty statement is the authoritative classification of the backwards compatibility of the change to the schema node in the specifed revision. This allows a human author to explicitly communicate the compatibilty and potentially override the rules specified in this document. This is useful in a number of situations including:

* When a tool may not be able to accurately determine the compatibilty of a change. For example, a change in a ’pattern’ or ’must’ statement can be difficult for a user or tool to determine if it is a compatible change.

* When a pattern, range or other statement is changed to more correctly define the server constraint. An example is correcting a pattern that incorrectly included 355.xxx.xxx.xxx as a possible IPv4 address to make it only accept up to 255.xxx.xxx.xxx.

Nothing about the backwards compatibility of a schema node is implied by the absence of a node compatibility statement. Hence, the schema node definition must be compared between the two revisions to determine the backwards compatibility.

If any nbc-change-at extension statements exists in a module or sub-module, then the module or sub-module MUST have non-backwards-compatible-revision substatements in each revision statement of the module or sub-module history where the revision matches the argument of any nbc-change-at statements. If any revision statements are removed, then all node compatibiilty statements that reference that revision MUST also be removed. Conversely, node compatibilty statements MUST NOT be removed unless the associated revision statement in the revision history is removed.

If a node compatiblity statement is added to a grouping, then all instances where the grouping is used in the module or by an importing module are also impacted by the compatibilty information. Similarly for a ’typedef’, all leafs and leaf-lists that use that typedef share the specified compatibility classification. A non-backwards-compatible change to a typedef or grouping defined in one module that is used by an importing module, does not cause the importing module to add a non-backwards-compatible-revision statement to the revision history. Non-backwards-compatible marking does not carry through import statements.

A node compatibility statement at a leaf, leaf-list, or typedef context takes precedence over a node compatibility statement in a typedef used by the leaf, leaf-list, or typedef. If multiple typedefs with compatibility statements are used by a leaf, leaf-list, or typedef (e.g. a union), and there is no compatibility statement at the top leaf, leaf-list, or typedef context, then the order of precedence used to classify the compatibility of the top level leaf, leaf-list, or typedef is as follows: nbc-change-at, bc-change-at, and finally editorial-change-at. That is, the leaf, leaf-list, or typedef takes the most impactful change classification of all the underlying typedefs.

Node compatibility statements are not supported on YANG statements such as ’pattern’ or ’range’. The compatibility statement instead goes against the leaf, leaf-list, or typedef context.

Node compatibility statements that refer to pre-release revisions of a module MUST be removed when a full release revision of the module is published.

Node compatibilty statements SHOULD NOT be used when it isn’t clear which change the statement is referring to. For example: If a leaf is reordered within a container, a node compatibility statement SHOULD NOT be used against the parent container nor against the reordered leaf. Similarly, if a leaf is renamed or moved to another context without keeping the old leaf present in the model and marked obsolete, a node compatibilty statement SHOULD not be used.

7. YANG module comparison algorithm

The schema comparison algorithm defined in Section 6 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.

8. YANG schema comparison algorithms

8.1. 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 7 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).

8.2. 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 6. 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.

9. 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.

10. 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@2023-02-14.yang"
module ietf-yang-rev-annotations {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-rev-annotations";
  prefix rev-ext;

  import ietf-yang-revisions {
    prefix rev;
  }

  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 2023-03-11 {
       rev:revision-label 1.0.0-draft-ietf-netmod-yang-schema-comparison-02;
       description
         "Draft revision";
       reference
         "XXXX: YANG Schema Comparison";
     }

     extension nbc-change-at {
       argument revision-date-or-label;
       description
         "A node compatibility statement that identifies a revision
         (by revision-label, or revision date if a revision-label is
         not available) where a non-backwards-compatible change has
         occurred in a particular YANG statement 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
         nbc-change-at substatements:
           - all schema node statements (leaf, rpc, choice, etc)
           - 'feature' statements
           - 'grouping' statements
           - 'identity' statements
           - 'must' statements
           - 'refine' statements
           - 'typedef' statements
           - YANG extensions

         Each YANG statement MUST only a have a single node
         compatibilty statement (one of nbc-change-at, bc-change-at,
         or editorial-change-at) for a particular revision. When a node
         has more than one of the node compatibilty statements (for
         different revisions), they must be ordered from most recent
         to least recent.

         An nbc-change-at statement can have 0 or 1 'description'
         substatements.

         The nbc-change-at statement in not inherited by descendants
         in the schema tree. It only applies to the specific YANG
         statement with which it is associated.
         ";

       reference
         "XXXX: YANG Schema Comparison;
          Section XXX, XXX";

     }

     extension bc-change-at {
       argument revision-date-or-label;
       description
         "A node compatibility statement that identifies a revision
         (by revision-label, or revision date if a revision-label is
         not available) where a backwards-compatible change has
         occurred in a particular YANG statement 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
         bc-change-at substatements:
           - all schema node statements (leaf, rpc, choice, etc)
           - 'feature' statements
           - 'grouping' statements
           - 'identity' statements
           - 'must' statements
           - 'refine' statements
           - 'typedef' statements
           - YANG extensions

         Each YANG statement MUST only a have a single node
         compatibilty statement (one of nbc-change-at, bc-change-at,
         or editorial-change-at) for a particular revision. When a node
         has more than one of the node compatibilty statements (for
         different revisions), they must be ordered from most recent
         to least recent.

         An bc-change-at statement can have 0 or 1 'description'
         substatements.

         The bc-change-at statement in not inherited by descendants
         in the schema tree. It only applies to the specific YANG
         statement with which it is associated.
         ";

       reference
         "XXXX: YANG Schema Comparison;
          Section XXX, XXX";

     }

     extension editorial-change-at {
       argument revision-date-or-label;
       description
         "A node compatibility statement that identifies a revision
         (by revision-label, or revision date if a revision-label is
         not available) where an editorial change has
         occurred in a particular YANG statement 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
         editorial-change-at substatements:
           - all schema node statements (leaf, rpc, choice, etc)
           - 'feature' statements
           - 'grouping' statements
           - 'identity' statements
           - 'must' statements
           - 'refine' statements
           - 'typedef' statements
           - YANG extensions

         Each YANG statement MUST only a have a single node
         compatibilty statement (one of nbc-change-at, bc-change-at,
         or editorial-change-at) for a particular revision. When a node
         has more than one of the node compatibilty statements (for
         different revisions), they must be ordered from most recent
         to least recent.

         An editorial-change-at statement can have 0 or 1 'description'
         substatements.

         The editorial-change-at statement in not inherited by descendants
         in the schema tree. It only applies to the specific YANG
         statement with which it is associated.
         ";

       reference
         "XXXX: YANG Schema Comparison;
          Section XXX, XXX";

     }

     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>
```

11. 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

* Jan Lindblad

* Per Andersson

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.

12. Security Considerations

The document does not define any new protocol or data model. There are no security impacts.

13. IANA Considerations

13.1. 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]

14. References

14.1. Normative References

[I-D.ietf-netmod-yang-module-versioning]
Wilton, R., Rahman, R., Lengyel, B., Clarke, J., and J. Sterne, "Updated YANG Module Revision Handling", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-module-versioning-08, 12 January 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-module-versioning-08>.

[I-D.ietf-netmod-yang-packages]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and B. Wu, "YANG Packages", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-packages-03, 4 March 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-packages-03>.

[I-D.ietf-netmod-yang-semver]
Clarke, J., Wilton, R., Rahman, R., Lengyel, B., Sterne, J., and B. Claise, "YANG Semantic Versioning", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-semver-10, 17 January 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-semver-10>.

[I-D.ietf-netmod-yang-solutions]
Wilton, R., "YANG Versioning Solution Overview", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-solutions-01, 2 November 2020, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-solutions-01>.

[I-D.ietf-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-versioning-reqs-07, 10 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-versioning-reqs-07>.

[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>.

[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>.

[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>.

14.2. Informative References

[I-D.clacla-netmod-yang-model-update]
Claise, B., Clarke, J., Lengyel, B., and K. D’Souza, "New YANG Module Update Procedure", Work in Progress, Internet-Draft, draft-clacla-netmod-yang-model-update-06, 2 July 2018, <https://datatracker.ietf.org/doc/html/draft-clacla-netmod-yang-model-update-06>.

Authors' Addresses

Per Andersson (editor)
Cisco Systems, Inc.
Email: perander@cisco.com

Robert Wilton
Cisco Systems, Inc.
Email: rwilton@cisco.com

Network Working Group
Internet-Draft
Updates: 8407 (if approved)
Intended status: Standards Track
Expires: 21 July 2023

J. Clarke, Ed.
R. Wilton, Ed.
Cisco Systems, Inc.
R. Rahman
Graphiant
B. Lengyel
Ericsson
J. Sterne
Nokia
B. Claise
Huawei
17 January 2023

YANG Semantic Versioning
draft-ietf-netmod-yang-semver-10

Abstract

This document specifies a scheme and guidelines for applying an extended set of semantic versioning rules to revisions of YANG artifacts (e.g., modules and packages). Additionally, this document defines an RFCAAAA-compliant revision-label-scheme for this YANG semantic versioning scheme.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

[I-D.ietf-netmod-yang-module-versioning] puts forth a number of concepts relating to modified rules for updating YANG modules and submodules, a means to signal when a new revision of a module or submodule has non-backwards-compatible (NBC) changes compared to its previous revision, and a scheme that uses the revision history as a lineage for determining from where a specific revision of a YANG module or submodule is derived. Additionally, section 3.4 of [I-D.ietf-netmod-yang-module-versioning] defines a revision-label which can be used as an alias to provide additional context or as a meaningful label to refer to a specific revision.

This document defines a revision-label scheme that uses extended semantic versioning rules [SemVer] for YANG artifacts (i.e., YANG modules, YANG submodules, and YANG packages [I-D.ietf-netmod-yang-packages]) as well as the revision label definition for using this scheme. The goal being to add a human readable revision label that provides compatibility information for the YANG artifact without needing to compare or parse its body. The label and rules defined herein represent the RECOMMENDED revision label scheme for IETF YANG artifacts.

Note that a specific revision of the SemVer 2.0.0 specification is referenced here (from June 19, 2020) to provide an immutable version. This is because the 2.0.0 version of the specification has changed over time without any change to the semantic version itself. In some cases the text has changed in non-backwards-compatible ways.

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 submodules, and YANG packages [I-D.ietf-netmod-yang-packages] are examples of YANG artifacts for the purposes of this document.

* SemVer: A version string that corresponds to the rules defined in [SemVer] . This specific camel-case notation is the one used by the SemVer 2.0.0 website and used within this document to distinguish between YANG Semver.

* YANG Semver: A revision-label identifier that is consistent with the extended set of semantic versioning rules, based on [SemVer] , defined within this document.

3. YANG Semantic Versioning

This section defines YANG Semantic Versioning, explains how it is used with YANG artifacts, and describes the rules associated with changing an artifact's semantic version when its contents are updated.

3.1. Relationship Between SemVer and YANG Semver

[SemVer] is completely compatible with YANG Semver in that a SemVer semantic version number is legal according to the YANG Semver rules (though the inverse is not necessarily true). YANG Semver is a superset of the SemVer rules, and allow for limited branching within YANG artifacts. If no branching occurs within a YANG artifact (i.e., you do not use the compatibility modifiers described below), the YANG Semver version label will appear as a SemVer version number.

3.2. YANG Semver Pattern

YANG artifacts that employ semantic versioning as defined in this document MUST use a version string (e.g., in revision-label or as a package version) that corresponds to the following pattern: 'X.Y.Z_COMPAT'. Where:

* X, Y and Z are mandatory non-negative integers that are each less than or equal to 2147483647 (i.e., the maximum signed 32-bit integer value) and MUST NOT contain leading zeroes,

* The '.' is a literal period (ASCII character 0x2e),

* The '_' is an optional single literal underscore (ASCII character 0x5f) and MUST only be present if the following COMPAT element is included,

* COMPAT, if specified, MUST be either the literal string "compatible" or the literal string "non_compatible".

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 YANG 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. While build metadata MUST be ignored when comparing YANG semantic versions, pre-release metadata MUST be used during module and submodule development as specified in Section 5 . Both pre-release and build metadata are allowed in order to support all the [SemVer] rules. Thus, a version lineage that follows strict [SemVer] rules is allowed for a YANG artifact.

To signal the use of this versioning scheme, modules and submodules MUST set the revision-label-scheme extension, as defined in [I-D.ietf-netmod-yang-module-versioning] , to the identity "yang-semver". That identity value is defined in the ietf-yang-semver module below.

Additionally, this ietf-yang-semver module defines a typedef that formally specifies the syntax of the YANG Semver.

3.3. Semantic Versioning Scheme for YANG Artifacts

This document defines the YANG semantic versioning scheme that is used for YANG artifacts that employ the YANG 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 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.ietf-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 and submodule versioned using the YANG semantic versioning scheme specifies the module's or submodule's semantic version as the argument to the 'rev:revision-label' statement.

Because the rules put forth in [I-D.ietf-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 or submodule may have been produced before this scheme was available.

YANG packages that make use of this YANG Semver will reflect that in the package metadata.

As stated above, the YANG semantic version is expressed as a string of the form: 'X.Y.Z_COMPAT'.

* '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 "_compatible" or "_non_compatible" modifier.

* 'Z' is the PATCH version. Changes in the PATCH version number can indicate an editorial change to the YANG artifact. In conjunction with the '_COMPAT' modifier (see below) changes to 'Z' may indicate a more substantive module 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. Some examples include correcting a spelling mistake in the description of a leaf within a YANG module or submodule, non-significant whitespace changes (e.g., realigning description statements or changing indentation), or changes to YANG comments. 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.

* '_COMPAT' is an additional modifier, unique to YANG Semver (i.e., not valid in [SemVer]), that indicates 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 '_COMPAT' takes:

  - If the modifier string is absent, the change represents an editorial change.

- If, however, the modifier string is present, the meaning is described below:

- "_compatible" - the change represents a backwards-compatible change

- "_non_compatible" - the change represents a non-backwards-compatible change

The '_COMPAT' modifier string is "sticky". Once a revision of a module has a modifier in the revision label, then all descendants of that revision with the same X.Y version digits will also have a modifier. The modifier can change from "_compatible" to "_non_compatible" in a descendant revision, but the modifier MUST NOT change from "_non_compatible" to "_compatible" and MUST NOT be removed. The persistence of the "_non_compatible" modifier ensures that comparisons of revision labels do not give the false impression of compatibility between two potentially non-compatible revisions. If "_non_compatible" was removed, for example between revisions "3.3.2_non_compatible" and "3.3.3" (where "3.3.3" was simply an editorial change), then comparing revision labels of "3.3.3" back to an ancestor "3.0.0" would look like they are backwards compatible when they are not (since "3.3.2_non_compatible" was in the chain of ancestors and introduced a non-backwards-compatible change).

The YANG artifact name and YANG semantic version uniquely identify a revision of said artifact. There MUST NOT be multiple instances of a YANG artifact definition with the same name and YANG semantic version but different content (and in the case of modules and submodules, 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 strings. E.g., artifact version "1.2.3_non_compatible" MUST NOT be defined if artifact version "1.2.3" has already been defined.

3.3.1. Branching Limitations with YANG Semver

YANG artifacts that use the YANG Semver revision-label scheme MUST ensure that two artifacts with the same MAJOR version number and no _compatible or _non_compatible modifiers are backwards compatible. Therefore, certain branching schemes cannot be used with YANG Semver. For example, the following branched parent-child module relationship using the following YANG Semver revision labels is not supported:

```
         3.5.0 -- 3.6.0 (add leaf foo)
           |
           |
         3.20.0 (added leaf bar)
```

In this case, given only the revision labels 3.6.0 and 3.20.0 without any parent-child relationship information, one would assume that 3.20.0 is backwards compatible with 3.6.0. But in the illegal example above, 3.20.0 is not backwards compatible with 3.6.0 since 3.20.0 does not contain the leaf foo.

Note that this type of branched parent-child relationship, where two revisions have different backwards compatible changes based on the same parent, is allowed in [I-D.ietf-netmod-yang-module-versioning] .

3.3.2. YANG Semver with submodules

YANG Semver MAY be used to version submodules. Submodule version are separate of any version on the including module, but if a submodule has changed, then the version of the including module MUST also be updated.

The rules for determining the version change of a submodule are the same as those defined in Section 3.2 and Section 3.3 as applied to YANG modules, except they only apply to the part of the module schema defined within the submodule's file.

One interesting case is moving definitions from one submodule to another in a way that does not change the resultant schema of the including module. In this case:

1. The including module has editorial changes

2. The submodule with the schema definition removed has non-backwards-compatible changes

3. The submodule with the schema definitions added has backwards-compatible changes

Note that the meaning of a submodule may change drastically despite having no changes in content or revision due to changes in other submodules belonging to the same module (e.g. groupings and typedefs declared in one submodule and used in another).

3.3.3. Examples for YANG semantic versions

The following diagram and explanation illustrate how YANG semantic versions work.

YANG Semantic versions for an example module:

```
         0.1.0
           |
         0.2.0
           |
         1.0.0
           |
         1.1.0 -> 1.1.1_compatible -> 1.1.2_non_compatible
           |
         1.2.0 -> 1.2.1_non_compatible -> 1.2.2_non_compatible
           |  \
         2.0.0 \
           |    \--> 1.3.0 -> 1.3.1_non_compatible
         3.0.0         |
           |         1.4.0
         3.1.0
```

The tree diagram above illustrates how the version history might evolve for an example module. The tree diagram only shows the parent/child ancestry relationships between the revisions. It does not describe the chronology of the revisions (i.e. when in time each revision was published relative to the other revisions).

The following description lists an example of what the chronological order of the revisions could look like, from oldest revision to newest:

0.1.0 - first pre-release module version

0.2.0 - second pre-release 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)

2.0.0 - change existing model for performance reasons, e.g. re-key list (NBC)

1.3.0 - improve existing functionality, added leaf "foo-64" (BC)

1.1.1_compatible - backport "foo-64" leaf to 1.1.x to avoid implementing "baz" from 1.2.0. This revision was created after 1.2.0 otherwise it may have been released as 1.2.0. (BC)

3.0.0 - NBC bugfix, rename "baz" to "bar"; also add new BC leaf "wibble"; (NBC)

1.3.1_non_compatible - backport NBC fix, rename "baz" to "bar" (NBC)

1.2.1_non_compatible - backport NBC fix, rename "baz" to "bar" (NBC)

1.1.2_non_compatible - NBC point bug fix, not required in 2.0.0 due to model changes (NBC)

1.4.0 - introduce new leaf "ghoti" (BC)

3.1.0 - introduce new leaf "wobble" (BC)

1.2.2_non_compatible - backport "wibble". This is a BC change but "non_compatible" modifier is sticky. (BC)

The partial ancestry relationships based on the semantic versioning numbers are as follows:

1.0.0 < 1.1.0 < 1.2.0 < 2.0.0 < 3.0.0 < 3.1.0

1.0.0 < 1.1.0 < 1.1.1_compatible < 1.1.2_non_compatible

1.0.0 < 1.1.0 < 1.2.0 < 1.2.1_non_compatible < 1.2.2_non_compatible

1.0.0 < 1.1.0 < 1.2.0 < 1.3.0 < 1.3.1_non_compatible

1.0.0 < 1.1.0 < 1.2.0 < 1.3.0 < 1.4.0

There is no ordering relationship between "1.1.1_non_compatible" and either "1.2.0" or "1.2.1_non_compatible", except that they share the common ancestor of "1.1.0".

Looking at the version number alone does not indicate ancestry. The module definition in "2.0.0", for example, does not contain all the contents of "1.3.0". Version "2.0.0" is not derived from "1.3.0".

3.4. YANG Semantic Version Update Rules

When a new revision of an artifact is produced, then the following rules define how the YANG semantic version for the new artifact revision is calculated, based on the changes between the two artifact revisions, and the YANG semantic version of the base artifact revision from which the changes are derived.

The following four rules specify the RECOMMENDED, and REQUIRED minimum, update to a YANG semantic version:

1. If an artifact is being updated in a non-backwards-compatible way, then the artifact version "X.Y.Z[_compatible|_non_compatible]" SHOULD be updated to "X+1.0.0" unless that version has already been used for this artifact but with different content, in which case the artifact version "X.Y.Z+1_non_compatible" SHOULD 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 SHOULD be updated to "X.Y+1.0", unless that version has already been used for this artifact but with different content, when the artifact version SHOULD be updated to "X.Y.Z+1_compatible" instead.

   ii "X.Y.Z_compatible" - the artifact version SHOULD be updated to "X.Y.Z+1_compatible".

   iii "X.Y.Z_non_compatible" - the artifact version SHOULD be updated to "X.Y.Z+1_non_compatible".

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 SHOULD be updated to "X.Y.Z+1"

   ii "X.Y.Z_compatible" - the artifact version SHOULD be updated to "X.Y.Z+1_compatible".

   iii "X.Y.Z_non_compatible" - the artifact version SHOULD be updated to "X.Y.Z+1_non_compatible".

4. YANG artifact semantic version numbers beginning with 0, i.e., "0.X.Y", are regarded as pre-release 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. See Section 5 for more details on using this notation during module and submodule development.

5. Additional pre-release rules for modules that have had at least one release are specified in Section 5 .

Although artifacts SHOULD be updated according to the rules above, which specify the recommended (and minimum required) update to the version number, the following rules MAY be applied when choosing a new version number:

1. An artifact author MAY update the version number with a more significant update than described by the rules above. For example, an artifact could be given a new MAJOR version number (i.e., X+1.0.0), even though no non-backwards-compatible changes have occurred, or an artifact could be given a new MINOR version number (i.e., X.Y+1.0) even if the changes were only editorial.

2. An artifact author MAY skip version numbers. That is, an artifact's revision history could be 1.0.0, 1.1.0, and 1.3.0 where 1.2.0 is skipped. Note that skipping versions has an impact when importing modules by revision-or-derived. See Section 4 for more details on importing modules with revision-label version gaps.

Although YANG Semver always indicates when a non-backwards-compatible, or backwards-compatible change may have occurred to a YANG artifact, it does not guarantee that such a change has occurred, or that consumers of that YANG artifact will be impacted by the change. Hence, tooling, e.g., [I-D.ietf-netmod-yang-schema-comparison] , also plays an important role for comparing YANG artifacts and calculating the likely impact from changes.

[I-D.ietf-netmod-yang-module-versioning] defines the "rev:non-backwards-compatible" extension statement to indicate where non-backwards-compatible changes have occurred in the module revision history. If a revision entry in a module's revision history includes the "rev:non-backwards-compatible" statement then that MUST be reflected in any YANG semantic version associated with that revision. However, the reverse does not necessarily hold, i.e., if the MAJOR version has been incremented it does not necessarily mean that a "rev:non-backwards-compatible" statement would be present.

3.5. Examples of the YANG Semver Label

3.5.1. 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 example-versioned-module {
       yang-version 1.1;
       namespace "urn:example:versioned:module";
       prefix "exvermod";
       rev:revision-label-scheme "ysver:yang-semver";

       import ietf-yang-revisions { prefix "rev"; }
       import ietf-yang-semver { prefix "ysver"; }

       description
         "to be completed";

       revision 2017-08-30 {
         description "Backport 'wibble' leaf";
         rev:revision-label 1.2.2_non_compatible;
       }

       revision 2017-07-30 {
         description "Rename 'baz' to 'bar'";
         rev:revision-label 1.2.1_non_compatible;
         rev:non-backwards-compatible;
       }

       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 2017-02-07 {
         description "First release version.";
         rev:revision-label 1.0.0;
       }

       // Note: YANG Semver rules do not apply to 0.X.Y labels.
       // The following pre-release revision statements would not
       // appear in any final published version of a module. They
       // are removed when the final version is published.
       // During the pre-release phase of development, only a
       // single one of these revision statements would appear

       // revision 2017-01-30 {
       //   description "NBC changes to initial revision";
       //   rev:revision-label 0.2.0;
       //   rev:non-backwards-compatible; // optional
       //                          // (theoretically no
       //                          // 'previous released version')
       // }

       // revision 2017-01-26 {
       //   description "Initial module version";
       //   rev:revision-label 0.1.0;
       // }

       //YANG module definition starts here
     }
```

3.5.2. Example of Package Using YANG Semver

Below is an example YANG package that uses the YANG Semver revision label based on the rules defined in this document. Note: '\' line wrapping per [RFC8792] .

```
   {
     "ietf-yang-instance-data:instance-data-set": {
       "name": "example-yang-pkg",
       "content-schema": {
         "module": "ietf-yang-packages@2022-03-04"
       },
       "timestamp": "2022-12-06T17:00:38Z",
       "description":  ["Example of a Package  \
          using YANG Semver"],
       "content-data": {
         "ietf-yang-packages:packages": {
           "package": [
             {
               "name": "example-yang-pkg",
               "version": "1.3.1",
               ...
             }
           ]
         }
       }
     }
   }
```

Figure 1

4. Import Module by Semantic Version

[I-D.ietf-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. The YANG Semver revision-label value can be used as the argument to rev:revision-or-derived . When used as such, any module that contains exactly the same YANG semantic version in its revision history may 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 "2.1.0_non_compatible" in order to determine if the present instance of module A derives from "2.0.0".

If an import by revision-or-derived cannot locate the specified revision-label in a given module's revision history, that import will fail. This is noted in the case of version gaps. That is, if a module's history includes "1.0.0", "1.1.0", and "1.3.0", an import from revision-or-derived at "1.2.0" will be unable to locate the specified revision entry and thus the import cannot be satisfied.

5. Guidelines for Using Semver During Module Development

This section and the IETF-specific sub-section below provides YANG Semver-specific guidelines to consider when developing new YANG modules. As such this section updates [RFC8407] .

Development of a brand new YANG module or submodule outside of the IETF that uses YANG Semver as its revision-label scheme SHOULD begin with a 0 for the MAJOR version component. This allows the module or submodule to disregard strict SemVer rules with respect to non-backwards-compatible changes during its initial development. However, module or submodule developers MAY choose to use the SemVer pre-release syntax instead with a 1 for the MAJOR version component. For example, an initial module or submodule revision-label might be either 0.0.1 or 1.0.0-alpha.1. If the authors choose to use the 0 MAJOR version component scheme, they MAY switch to the pre-release scheme with a MAJOR version component of 1 when the module or submodule is nearing initial release (e.g., a module's or submodule's revision label may transition from 0.3.0 to 1.0.0-beta.1 to indicate it is more mature and ready for testing).

When using pre-release notation, the format MUST include at least one alphabetic component and MUST end with a '.' or '-' and then one or more digits. These alphanumeric components will be used when deciding pre-release precedence. The following are examples of valid pre-release versions:

1.0.0-alpha.1

1.0.0-alpha.3

2.1.0-beta.42

3.0.0-202007.rc.1

When developing a new revision of an existing module or submodule using the YANG Semver revision-label scheme, the intended target semantic version MUST be used along with pre-release notation. For example, if a released module or submodule which has a current revision-label of 1.0.0 is being modified with the intent to make non-backwards-compatible changes, the first development MAJOR version component must be 2 with some pre-release notation such as -alpha.1, making the version 2.0.0-alpha.1. That said, every publicly available release of a module or submodule MUST have a unique YANG Semver revision-label (where a publicly available release is one that could be implemented by a vendor or consumed by an end user). Therefore, it may be prudent to include the year or year and month development began (e.g., 2.0.0-201907-alpha.1). As a module or submodule undergoes development, it is possible that the original intent changes. For example, a 1.0.0 version of a module or submodule that was destined to become 2.0.0 after a development cycle may have had a scope change such that the final version has no non-backwards-compatible changes and becomes 1.1.0 instead. This change is acceptable to make during the development phase so long as pre-release notation is present in both versions (e.g., 2.0.0-alpha.3 becomes 1.1.0-alpha.4). However, on the next development cycle (after 1.1.0 is released), if again the new target release is 2.0.0, new pre-release components must be used such that every revision-label for a given module or submodule MUST be unique throughout its entire lifecycle (e.g., the first pre-release version might be 2.0.0-202005-alpha.1 if keeping the same year and month notation mentioned above).

5.1. Pre-release Version Precedence

As a module or submodule is developed, the scope of the work may change. That is, while a ratified module or submodule with revision-label 1.0.0 is initially intended to become 2.0.0 in its next ratified version, the scope of work may change such that the final version is 1.1.0. During the development cycle, the pre-release versions could move from 2.0.0-some-pre-release-tag to 1.1.0-some-pre-release-tag. This downwards changing of version numbers makes it difficult to evaluate semantic version rules between pre-release versions. However, taken independently, each pre-release version can be compared to the previously ratified version (e.g., 1.1.0-some-pre-release-tag and 2.0.0-some-pre-release-tag can each be compared to 1.0.0). Module and submodule developers SHOULD maintain only one revision statement in a pre-released module or submodule that reflects the latest revision. IETF authors MAY choose to include an appendix in the associated draft to track overall changes to the module or submodule.

5.2. YANG Semver in IETF Modules

All published IETF modules and submodules MUST use YANG semantic versions for their revision-labels.

Development of a new module or submodule within the IETF SHOULD begin with the 0 MAJOR number scheme as described above. When revising an existing IETF module or submodule, the revision-label MUST use the target (i.e., intended) MAJOR and MINOR version components with a 0 PATCH version component. If the intended ratified release will be non-backward-compatible with the current ratified release, the MINOR version component MUST be 0.

5.2.1. Guidelines for IETF Module Development

All IETF modules and submodules in development MUST use the whole document name as a pre-release version string, including the current document revision. For example, if a module or submodule which is currently released at version 1.0.0 is being revised to include non-backwards-compatible changes in draft-user-netmod-foo, its development revision-labels MUST include 2.0.0-draft-user-netmod-foo followed by the document’s revision (e.g., 2.0.0-draft-user-netmod-foo-02). This will ensure each pre-release version is unique across the lifecycle of the module or submodule. Even when using the 0 MAJOR version for initial module or submodule development (where MINOR and PATCH can change), appending the draft name as a pre-release component helps to ensure uniqueness when there are perhaps multiple, parallel efforts creating the same module or submodule.

Some draft revisions may not include an update to the YANG modules or submodules contained in the draft. In that case, those modules or submodules that are not updated do not not require a change to their versions. Updates to the YANG Semver version MUST only be done when the revision of the module changes.

See Appendix A for a detailed example of IETF pre-release versions.

5.2.2. Guidelines for Published IETF Modules

For IETF YANG modules and submodules that have already been published, revision-labels MUST be retroactively applied to all existing revisions when the next new revision is created, starting at version "1.0.0" for the initial published revision, and then incrementing according to the YANG Semver version rules specified in Section 3.4 . For example, if a module or submodule started out in the pre-NMDA ([RFC8342]) world, and then had NMDA support added without removing any legacy "state" branches -- and you are looking to add additional new features -- a sensible choice for the target YANG Semver would be 1.2.0 (since 1.0.0 would have been the initial, pre-NMDA release, and 1.1.0 would have been the NMDA revision).

6. YANG Module

This YANG module contains the typedef for the YANG semantic version and the identity to signal its use.

```
<CODE BEGINS> file "ietf-yang-semver@2023-01-17.yang"
module ietf-yang-semver {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-semver";
  prefix ysver;
  rev:revision-label-scheme "yang-semver";

  import ietf-yang-revisions {
    prefix rev;
  }

  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>
     Author:   Robert Wilton
               <mailto:rwilton@cisco.com>
```

```
        Author:   Reshad Rahman
                  <mailto:reshad@yahoo.com>
        Author:   Balazs Lengyel
                  <mailto:balazs.lengyel@ericsson.com>
        Author:   Jason Sterne
                  <mailto:jason.sterne@nokia.com>
        Author:   Benoit Claise
                  <mailto:benoit.claise@huawei.com>";
     description
       "This module provides type and grouping definitions for YANG
        packages.

        Copyright (c) 2022 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 Revised 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.
     // RFC Ed. update the rev:revision-label to "1.0.0".

     revision 2023-01-17 {
       rev:label "1.0.0-draft-ietf-netmod-yang-semver-10";
       description
         "Initial revision";
       reference
         "RFC XXXX: YANG Semantic Versioning.";
     }

     /*
      * Identities
      */

     identity yang-semver {
       base rev:revision-label-scheme-base;
       description
         "The revision-label scheme corresponds to the YANG Semver
          scheme which is defined by the pattern in the 'version'
          typedef below. The rules governing this revision-label
          scheme are defined in the reference for this identity.";
       reference
         "RFC XXXX: YANG Semantic Versioning.";
     }

     /*
      * Typedefs
      */

     typedef version {
       type rev:revision-label {
         pattern '[0-9]+[.][0-9]+[.][0-9]+(_(non_)?compatible)?'
               + '(-[A-Za-z0-9.-]+[.-][0-9]+)?([+][A-Za-z0-9.-]+)?';
       }
       description
         "Represents a YANG semantic version.  The rules governing the
          use of this revision label scheme are defined in the
          reference for this typedef.";
       reference
         "RFC XXXX: YANG Semantic Versioning.";
     }
   }
   <CODE ENDS>
```

7. 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, Bo Wu, Ebben Aries, Jan Lindblad, Jason Sterne, Joe Clarke, Juergen Schoenwaelder, Mahesh Jethanandani, Michael (Wangzitao), Qin Wu, Reshad Rahman, and Rob Wilton.

The initial revision of this document was refactored and built upon [I-D.clacla-netmod-yang-model-update] . We would like the thank Kevin D'Souza for his initial work in this problem space.

Discussions 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.

8. 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.

That said, the YANG module in this document does not define any schema nodes (i.e., nothing that can be read or written). It only defines a typedef and an identity. Therefore, there is no need to further protect any nodes with access control.

9. IANA Considerations

9.1. YANG Module Registrations

This document requests IANA to register a URI in the "IETF XML Registry" [RFC3688] . Following the format in RFC 3688, the following registration is requested.

URI: urn:ietf:params:xml:ns:yang:ietf-yang-semver

Registrant Contact: The IESG.

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

The following YANG module is requested to be registered in the "IANA Module Names" [RFC6020] . Following the format in RFC 6020, the following registrations are requested:

The ietf-yang-semver module:

Name: ietf-yang-semver

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-semver

Prefix: ysver

Reference: [RFCXXXX]

9.2. Guidance for YANG Semver in IANA maintained YANG modules and submodules

Note for IANA (to be removed by the RFC editor): Please check that the registries and IANA YANG modules and submodules are referenced in the appropriate way.

IANA is responsible for maintaining and versioning some YANG modules and submodules, e.g., iana-if-types.yang [IfTypeYang] and iana-routing-types.yang [RoutingTypesYang] .

In addition to following the rules specified in the IANA Considerations section of [I-D.ietf-netmod-yang-module-versioning] , IANA maintained YANG modules and submodules MUST also include a YANG Semver revision label for all new revisions, as defined in Section 3 .

The YANG Semver version associated with the new revision MUST follow the rules defined in Section 3.4 .

Note: For IANA maintained YANG modules and submodules that have already been published, revision labels MUST be retroactively applied to all existing revisions when the next new revision is created, starting at version "1.0.0" for the initial published revision, and then incrementing according to the YANG Semver rules specified in Section 3.4 .

Most changes to IANA maintained YANG modules and submodules are expected to be backwards-compatible changes and classified as MINOR version changes. The PATCH version may be incremented instead when only editorial changes are made, and the MAJOR version would be incremented if non-backwards-compatible changes are made.

Given that IANA maintained YANG modules are versioned with a linear history, it is anticipated that it should not be necessary to use the "_compatible" or "_non_compatible" modifiers to the "Z_COMPAT" version element.

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>.

[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>.

[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>.

[I-D.ietf-netmod-yang-module-versioning] Wilton, R., Rahman, R., Lengyel, B., Clarke, J., and J. Sterne, "Updated YANG Module Revision Handling", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-module-versioning-08, 12 January 2023, <https://www.ietf.org/archive/id/draft-ietf-netmod-yang-module-versioning-08.txt>.

10.2. Informative References

[I-D.clacla-netmod-yang-model-update] Claise, B., Clarke, J., Lengyel, B., and K. D’Souza, "New YANG Module Update Procedure", Work in Progress, Internet-Draft, draft-clacla-netmod-yang-model-update-06, 2 July 2018, <https://www.ietf.org/archive/id/draft-clacla-netmod-yang-model-update-06.txt>.

[I-D.ietf-netmod-yang-packages] Wilton, R., Rahman, R., Clarke, J., Sterne, J., and B. Wu, "YANG Packages", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-packages-03, 4 March 2022, <https://www.ietf.org/archive/id/draft-ietf-netmod-yang-packages-03.txt>.

[I-D.ietf-netmod-yang-schema-comparison] Wilton, R., "YANG Schema Comparison", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-schema-comparison-01, 2 November 2020, <https://www.ietf.org/archive/id/draft-ietf-netmod-yang-schema-comparison-01.txt>.

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/info/rfc8792>.

[openconfigsemver]
"Semantic Versioning for Openconfig Models", <http://www.openconfig.net/docs/semver/>.

[SemVer] "Semantic Versioning 2.0.0 (text from June 19, 2020)", <https://github.com/semver/semver/blob/8b2e8eec394948632957639dfa99fc7ec6286911/semver.md>.

[IfTypeYang]
"iana-if-type YANG Module", <https://www.iana.org/assignments/iana-if-type/iana-if-type.xhtml>.

[RoutingTypesYang]
"iana-routing-types YANG Module", <https://www.iana.org/assignments/iana-routing-types/iana-routing-types.xhtml>.

Appendix A. Example IETF Module Development

Assume a new YANG module is being developed in the netmod working group in the IETF. Initially, this module is being developed in an individual internet draft, draft-jdoe-netmod-example-module. The following represents the initial version tree (i.e., value of revision-label) of the module as it's being initially developed.

Version lineage for initial module development:

```
      0.0.1-draft-jdoe-netmod-example-module-00
        |
      0.1.0-draft-jdoe-netmod-example-module-01
        |
      0.2.0-draft-jdoe-netmod-example-module-02
        |
      0.2.1-draft-jdoe-netmod-example-module-03
```

At this point, development stabilizes, and the workgroup adopts the draft. Thus now the draft becomes draft-ietf-netmod-example-module. The initial pre-release lineage continues as follows.

Continued version lineage after adoption:

```
    1.0.0-draft-ietf-netmod-example-module-00
      |
    1.0.0-draft-ietf-netmod-example-module-01
      |
    1.0.0-draft-ietf-netmod-example-module-02
```

At this point, the draft is ratified and becomes RFC12345 and the YANG module version becomes 1.0.0.

A time later, the module needs to be revised to add additional capabilities. Development will be done in a backwards-compatible way. Two new individual drafts are proposed to go about adding the capabilities in different ways: draft-jdoe-netmod-exmod-enhancements and draft-asmith-netmod-exmod-changes. These are initially developed in parallel with the following versions.

Parallel development for next module revision (track 1):

```
      1.1.0-draft-jdoe-netmod-exmod-enhancements-00
        |
      1.1.0-draft-jdoe-netmod-exmod-enhancements-01
```

In parallel with (track 2):

```
      1.1.0-draft-asmith-netmod-exmod-changes-00
        |
      1.1.0-draft-asmith-netmod-exmod-changes-01
```

At this point, the WG decides to merge some aspects of both and adopt the work in asmith's draft as draft-ietf-netmod-exmod-changes. A single version lineage continues.

```
      1.1.0-draft-ietf-netmod-exmod-changes-00
        |
      1.1.0-draft-ietf-netmod-exmod-changes-01
        |
      1.1.0-draft-ietf-netmod-exmod-changes-02
        |
      1.1.0-draft-ietf-netmod-exmod-changes-03
```

The draft is ratified, and the new module version becomes 1.1.0.

Authors' Addresses

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

Robert Wilton (editor)
Cisco Systems, Inc.
Email: rwilton@cisco.com

Reshad Rahman
Graphiant
Email: reshad@yahoo.com

Balazs Lengyel
Ericsson
1117 Budapest
Magyar Tudosok Korutja
Hungary
Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

Jason Sterne
Nokia
Email: jason.sterne@nokia.com

Benoit Claise
Huawei
Email: benoit.claise@huawei.com

NETMOD
Internet-Draft
Intended status: Standards Track
Expires: 10 September 2023

Q. Ma
Q. Wu
Huawei
B. Lengyel
Ericsson
H. Li
HPE

9 March 2023

YANG Extension and Metadata Annotation for Immutable Flag
draft-ma-netmod-immutable-flag-05

Abstract

This document defines a way to formally document as a YANG extension or YANG metadata an existing model handling behavior: modification restrictions on data declared as configuration.

This document defines a YANG extension named "immutable" to indicate that specific "config true" data nodes are not allowed to be created/deleted/updated. To indicate that specific entries of a list/leaf-list node or instances inside list entries cannot be updated/deleted after initialization, a metadata annotation with the same name is also defined. Any data node or instance marked as immutable is read-only to the clients of YANG-driven management protocols, such as NETCONF, RESTCONF and other management operations (e.g., SNMP and CLI requests).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines a way to formally document as a YANG extension or YANG metadata an existing model handling behavior that is already allowed in YANG and which has been used by multiple standard organizations and vendors. It is the aim to create one single standard solution for documenting modification restrictions on data declared as configuration, instead of the multiple existing vendor and organization specific solutions. See Appendix Bfor existing implementations.

YANG [RFC7950] is a data modeling language used to model both state and configuration data, based on the "config" statement. However there exists data that cannot be modified by the client(it is immutable), but still needs to be declared as "config true" to:

* allow configuration of data nodes under immutable lists or containers;

* place "when", "must" and "leafref" constraints between configuration and immutable schema nodes.

* ensure the existence of specific list entries that are provided and needed by the system, while additional list entries can be created, modified or deleted;

Clients believe that "config true" nodes are modifiable even though the server is allowed to reject such a modification at any time. If the server knows that it will reject the modification, it should document this towards the clients in a machine readable way.

To address this issue, this document defines a YANG extension named "immutable" to indicate that specific "config true" data nodes are not allowed to be created/deleted/updated. To indicate that specific entries of a list/leaf-list node or instances inside list entries cannot be updated/deleted after initialization, a metadata annotation [RFC7952] with the same name is also defined. Any data node or instance marked as immutable is read-only to the clients of YANG-driven management protocols, such as NETCONF, RESTCONF and other management operations (e.g., SNMP and CLI requests). Marking instance data nodes as immutable (as opposed to marking schema-nodes) is useful when only some instances of a list or leaf-list shall be marked as read-only.

Immutability is an existing model handling practice. While in some cases it is needed, it also has disadvantages, therefore it SHOULD be avoided wherever possible.

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

UC1 Modeling of server capabilities

UC2 HW based autoconfiguration

UC3 Predefined Access control Rules

UC4 Declaring System defined configuration unchangeable

UC5 Immutable BGP AS number and peer type

UC6 Modeling existing data handling behavior in other standard organizations

Appendix A describes the use cases in detail.

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.

The following terms are defined in [RFC6241] and [RFC8341] and are not redefined here:

* configuration data

* access operation

* write access

The following terms are defined in this document:

immutable: A schema or instance node property indicating that the configuration data is not allowed to be created/deleted/updated.

1.2. Applicability

The "immutable" concept defined in this document only indicates write access restrictions to writable datastores. A particular data node or instance MUST have the same immutability in all writable datastores. The immutable annotation information should also be visible in read-only datastores (e.g., <system>, <intended>, <operational>), however this only serves as information about the data node itself, but has no effect on the handling of the read-only datastore.

The immutability property of a particular data node or instance MUST be protocol-independent and user-independent.

2. Solution Overview

Already some servers handle immutable configuration data and will reject any attempt to the "create", "delete" or "update" such data. This document allows the existing immutable data node or instance to be marked by YANG extension or metadata annotation. Requests to create/update/delete an immutable configuration data always return an error (if no corresponding "exceptions" are declared in a YANG extension). The error reporting is performed immediately at an <edit-config> operation time, regardless what the target configuration datastore is. For an example of an "invalid-value" error response, see Appendix A.2.1.

However, the following operations SHOULD be allowed for immutable nodes:

* Use a create, update, delete/remove operation on an immutable node if the effective change is null. E.g., if a leaf has a current value of "5" it should be allowed to replace it with a value of "5";

* Create an immutable data node with a same value that already exists in the <system> datastore.;

Note that even if a particular data node is immutable without the exception for "delete", it still can be deleted if its parent node is deleted, e.g., /if:interfaces/if:interface/if:type leaf is immutable, but the deletion to the /if:interfaces/if:interface list entry is allowed; if a particular data node is immutable without the exception for "create", it means the client can never create the instance of it, regardless the handling of its parent node; it may be created by the system or have a default value when its parent is created.

In some cases adding the immutable property is allowed but does not have any additional semantic meaning. For example, a key leaf is given a value when a list entry is created, and cannot be modified and deleted unless the list entry is deleted. A mandatory leaf MUST exist and cannot be deleted if the ancestor node exists in the data tree.

3. "Immutable" YANG Extension

3.1. Definition

The "immutable" YANG extension can be a substatement to a "config true" leaf, leaf-list, container, list, anydata or anyxml statement. It has no effect if used as a substatment to a "config false" node, but can be allowed anyway. When present, it indicates that data nodes based on the parent statement are not allowed to be added, removed or updated except according to the exceptions argument. Any such write attempt will be rejected by the server.

The "immutable" YANG extension defines an argument statement named "exceptions" which gives a list of operations that users are permitted to invoke for the specified node.

The following values are supported for the "exceptions" argument:

* create: allow users to create instances of the data node;

* update: allow users to modify instances of the data node;

* delete: allow users to delete instances of the data node.

If more than one value is used, a space-separated string for the "exceptions" argument is used. For example, if a particular data node can be created and modified, but cannot be deleted, the following "immutable" YANG extension with "create" and "update" exceptions should be defined in a substatement to that data node:

```
immutable "create update";
```

Providing an empty string for the "exceptions" argument is equivalent to a single extension without an argument followed. Providing all 3 values can be used to override immutability inherited from its ancestor node. For data nodes with no write access restriction inherited from its ancestor node (see Section 3.2), providing all 3 values has the same effect as not using this extension at all, but can be used anyway.

Note that leaf-list instances can be created and deleted, but not modified. Any exception for "update" operation to leaf-list data nodes SHALL be ignored.

3.2. Inheritance of Immutable YANG Extension

Immutability specified by the use of the 'immutable' extension statement (including any exception argument) is inherited by all child and descendant nodes of a container or a list. It is possible to override thhe inherited immutability property by placing another immuable extension statement on a specific child/descendant node. For example, given the following list definition:

```
list application {
  im:immutable "create delete";
  key name;
  leaf name {
    type string;
  }
  leaf protocol {
    im:immutable;
    type enumeration {
      enum tcp;
      enum udp;
    }
  }
  leaf port-number {
    im:immutable "create update delete";
    type int16;
  }
}
```

application list entries are allowed to be created and deleted, but cannot be modified; "protocol" cannot be changed in any way while "port-number" can be created, modified or deleted. Using the immutable statement with exception argument we can make immutability stricter (for the protocol child node) or less restrictive (for the port-number child node).

4. "Immutable" Metadata Annotation

4.1. Definition

The "immutable" flag SHALL be used to indicate the immutability of a particular instantiated data node. It can only be used for list/ leaf-list entries. The "immutable" flag is of type boolean.

Note that "immutable" metadata annotation is used to annotate instances of a list/leaf-list rather than schema nodes. A list may have multiple entries/instances in the data tree, "immutable" can annotate some of the instances as read-only, while others are read-write.

Any list/leaf-list instance annotated with immutable="true" by the server is read-only to clients and cannot be updated/deleted. If a list entry is annotated with immutable="true", the whole instance is read-only and any contained descendant configuration is not allowed to be created, updated and deleted. Descendant nodes SHALL NOT carry the immutable annotation.

When the client retrieves data from a particular datastore, immutable data node instances MUST be annotated with immutable="true" by the server. If the "immutable" metadata annotation for a list/leaf-list entry is not specified, the default "immutable" value is false. Explicitly annotating instances as immutable="false" has the same effect as not specifying this value.

5. Interaction between Immutable YANG Extension and Metadata Annotation

When a client reads data from a datastore, if a data node is specified as immutable using the extension statement, the corresponding data node instances generally SHALL NOT be marked with the immutable annotation. However, if the immutable extension statement has exceptions defined, the server MAY decide that for a particular list entriy or leaf-list instance strict immutability shall apply without exceptions. In this case the server SHALL mark the relevant data node instances with the immutable annotation. The immutable annotation overrides any exceptions specified for the immutabile statement inlcuding any exception on any descendant nodes.

6. Interaction between Immutable Flag and NACM

If a data node or some list or leaf-list entries are immutable the server MUST reject any operation that attempts to create, delete or update them, however the "exceptions" argument, if present, SHALL be taken into account. Rejecting an operation due to immutability SHALL be done indepent of any access control settings.

7. YANG Module

```
   <CODE BEGINS>
    file="ietf-immutable@2022-12-14.yang"
   //RFC Ed.: replace XXXX with RFC number and remove this note
     module ietf-immutable {
       yang-version 1.1;
       namespace "urn:ietf:params:xml:ns:yang:ietf-immutable";
       prefix im;

       import ietf-yang-metadata {
         prefix md;
       }

       organization
         "IETF Network Modeling (NETMOD) Working Group";

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

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

          Author: Qiufang Ma
                  <mailto:maqiufang1@huawei.com>

          Author: Qin Wu
                  <mailto:bill.wu@huawei.com>

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

          Author: Hongwei Li
                  <mailto:flycoolman@gmail.com>";

       description
         "This module defines a metadata annotation named 'immutable'
          to indicate the immutability of a particular instantiated
          data node. Any instantiated data node marked with
          immutable='true' by the server is read-only to the clients
          of YANG-driven management protocols, such as NETCONF,
          RESTCONF as well as SNMP and CLI requests.

          The module defines the immutable extension that indicates
          that data nodes based on the parent data-definition
          statement cannot be created, removed, or updated
          except according to the 'exceptions' argument.

          Copyright (c) 2022 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 Revised
          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 HHHH
          (https://www.rfc-editor.org/info/rfcHHHH); 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 2022-12-14 {
         description
           "Initial revision.";
         // RFC Ed.: replace XXXX and remove this comment
         reference
           "RFC XXXX: YANG Extension and Metadata Annotation for
            Immutable Flag";
       }

       extension immutable {
         argument exceptions;
         description
           "The 'immutable' extension as a substatement to a data
            definition statement indicates that data nodes based on
            the parent statement MUST NOT be added, removed or
            updated by management protocols, such as NETCONF,
            RESTCONF or other management operations (e.g., SNMP
            and CLI requests) except when indicated by the
            exceptions argument.

            Immutable data MAY be marked as config true to allow
            'leafref', 'when' or 'must' constraints to be based
            on it.

            The statement MUST only be a substatement of the leaf,
            leaf-list, container, list, anydata, anyxml statements.
            Zero or one immutable statement per parent statement
            is allowed.
            No substatements are allowed.
            The argument is a list of space-separated operations that
            are permitted to be used for the specified node, while
            other operations are forbidden by the immutable extension.
            - create: allows users to create instances of the data node
            - update: allows users to modify instances of the data node
            - delete: allows users to delete instances of the data node

            To disallow all user write access, omit the argument;

            To allow only create and delete user access, provide
            the string 'create delete' for the 'exceptions' parameter.

            Equivalent YANG definition for this extension:

            leaf immutable {
              type bits {
                bit create;
                bit update;
                bit delete;
              }
              default '';
            }

            Immutability specified by the use of the 'immutable' extension
            statement (including any exception argument) is inherited by all
            child and descendant nodes of a container or a list. It is possible
            to override the inherited immutability property by placing another
            immutable extension statement on a specific child/descendant node.

            Adding immutable or removing values from the
            exceptions argument of an existing immutable statement
            are non-backwards compatible changes.
            Other changes to immutable are backwards compatible.";
       }

       md:annotation immutable {
         type boolean;
         description
           "The 'immutable' annotation indicates the immutability of an
            instantiated data node. Any data node instance marked as
            'immutable=true' is read-only to clients and cannot be
            updated through NETCONF, RESTCONF or CLI. It applies to the
            list and leaf-list entries. If a list entry is annotated
            with immutable='true', the whole instance is read-only and
            including any contained descendant data nodes.
            The default is 'immutable=false' if not specified for an instance.";
       }
     }
```

<CODE ENDS>

8. IANA Considerations

8.1. The "IETF XML" Registry

This document registers one XML namespace URN in the 'IETF XML registry', following the format defined in [RFC3688].

URI: urn:ietf:params:xml:ns:yang:ietf-immutable
Registrant Contact: The IESG.
XML: N/A, the requested URIs are XML namespaces.

8.2. The "YANG Module Names" Registry

This document registers one module name in the 'YANG Module Names' registry, defined in [RFC6020].

name: ietf-immutable
prefix: im
namespace: urn:ietf:params:xml:ns:yang:ietf-immutable
RFC: XXXX
// RFC Ed.: replace XXXX and remove this comment

9. Security Considerations

The YANG module specified in this document defines a YANG extension and a metadata Annotation. These can be used to further restrict write access but cannot be used to extend access rights.

This document does not define any protocol-accessible data nodes.

Since immutable information is tied to applied configuration values, it is only accessible to clients that have the permissions to read the applied configuration values.

The security considerations for the Defining and Using Metadata with YANG (see Section 9 of [RFC7952]) apply to the metadata annotation defined in this document.

Acknowledgements

Thanks to Kent Watsen, Andy Bierman, Robert Wilton, Jan Lindblad, Reshad Rahman, Anthony Somerset, Lou Berger, Joe Clarke, Scott Mansfield for reviewing, and providing important input to, this document.

References

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>.

[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>.

[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>.

[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>.

Informative References

[I-D.ietf-netmod-system-config]
Ma, Q., Wu, Q., and C. Feng, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ietf-netmod-system-config-01, 4 January 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-system-config-01>.

[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>.

[TR-531] ONF, "UML to YANG Mapping Guidelines, <https://wiki.opennetworking.org/download/attachments/376340494/Draft_TR-531_UML-YANG_Mapping_Gdls_v1.1.03.docx?version=5&modificationDate=1675432243513&api=v2>", February 2023.

[TS28.623] 3GPP, "Telecommunication management; Generic Network Resource Model (NRM) Integration Reference Point (IRP); Solution Set (SS) definitions, <https://www.3gpp.org/ftp/Specs/archive/28_series/28.623/28623-i02.zip>".

[TS32.156] 3GPP, "Telecommunication management; Fixed Mobile Convergence (FMC) Model repertoire, <https://www.3gpp.org/ftp/Specs/archive/32_series/32.156/32156-h10.zip>".

Appendix A. Detailed Use Cases

A.1. UC1 - Modeling of server capabilities

System capabilities might be represented as system-defined data nodes in the model. Configurable data nodes might need constraints specified as "when", "must" or "path" statements to ensure that configuration is set according to the system's capabilities. E.g.,

* A timer can support the values 1,5,8 seconds. This is defined in the leaf-list 'supported-timer-values'.

* When the configurable 'interface-timer' leaf is set, it should be ensured that one of the supported values is used. The natural solution would be to make the 'interface-timer' a leaf-ref pointing at the 'supported-timer-values'.

However, this is not possible as 'supported-timer-values' must be read-only thus config=false while 'interface-timer' must be writable thus config=true. According to the rules of YANG it is not allowed to put a constraint between config true and false schema nodes.

The solution is that the supported-timer-values data node in the YANG Model shall be defined as "config true" and shall also be marked with the "immutable" extension making it unchangable. After this the 'interface-timer' shall be defined as a leaf-ref pointing at the 'supported-timer-values'.

A.2. UC2 - HW based autoconfiguration - Interface Example

This section shows how to use immutable YANG extension to mark some data node as immutable.

When an interface is physically present, the system will create an interface entry automatically with valid name and type values in <system> (if exists, see [I-D.ietf-netmod-system-config]). The system-generated data is dependent on and must represent the HW present, and as a consequence must not be changed by the client. The data is modelled as "config true" and should be marked as immutable.

Seemingly an alternative would be to model the list and these leaves as "config false", but that does not work because:

* The list cannot be marked as "config false", because it needs to contain configurable child nodes, e.g., ip-address or enabled;

* The key leaf (name) cannot be marked as "config false" as the list itself is config true;

* The type cannot be marked "config false", because we MAY need to reference the type to make different configuration nodes conditionally available.

The immutability of the data is the same for all interface instances, thus following fragment of a fictional interface module including an "immutable" YANG extension can be used:

```
        container interfaces {
          list interface {
            key "name";
            leaf name {
              type string;
            }
            leaf type {
              im:immutable "create delete";
              type identityref {
                base ianaift:iana-interface-type;
              }
              mandatory true;
            }
            leaf mtu {
              type uint16;
            }
            leaf-list ip-address {
              type inet:ip-address;
            }
          }
        }
```

Note that the "name" leaf is defined as a list key which can never been modified for a particular list entry, there is no need to mark "name" as immutable.

A.2.1. Error Response to Client Updating the Value of an Interface Type

This section shows an example of an error response due to the client modifying an immutable configuration.

Assume the system creates an interface entry named "eth0" given that an inerface is inserted into the device. If a client tries to change the type of an interface to a value that doesn’t match the real type of the interface used by the system, the request will be rejected by the server:

```
   <rpc message-id="101"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
        xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
     <edit-config>
       <target>
         <running/>
       </target>
       <config>
         <interface xc:operation="merge"
               xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
           <name>eth0</name>
           <type>ianaift:tunnel</type>
         </interface>
       </config>
     </edit-config>
   </rpc>

   <rpc-reply message-id="101"
              xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
              xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
     <rpc-error>
       <error-type>application</error-type>
       <error-tag>invalid-value</error-tag>
       <error-severity>error</error-severity>
       <error-path xmlns:t="http://example.com/schema/1.2/config">
         /interfaces/interface[name="eth0"]/type
       </error-path>
       <error-message xml:lang="en">
         Invalid type for interface eth0
       </error-message>
     </rpc-error>
   </rpc-reply>
```

A.3. UC3 - Predefined Access control Rules

Setting up detailed rules for access control is a complex task. (see [RFC8341]) A vendor may provide an initial, predefined set of groups and related access control rules so that the customer can use access control out-of-the-box. The customer may continue using these predefined rules or may add his own groups and rules. The predefined groups shall not be removed or altered guaranteeing that access control remains usable and basic functions e.g., a system-security-administrator are always available.

The system needs to protect the predefined groups and rules, however, the list "groups" or the list "rule-list" cannot be marked as config=false or with the "immutable" extension in the YANG model because that would prevent the customer adding new entries. Still it would be good to notify the client in a machine readable way that the predefined entries cannot be modified. When the client retrieves access control data the immutable="true" metadata annotation should be used to indicate to the client that the predefined groups and rules cannot be modified.

A.4. UC4 - Declaring System defined configuration unchangeable

As stated in [I-D.ietf-netmod-system-config] the device itself might supply some configuration. As defined in that document in section "5.4. Modifying (overriding) System Configuration" the server may allow some parts of system configuration to be modified while other parts of the system configuration are non-modifiable. The immutable extension or metadata annotation can be used to define which parts are non-modifiable and to inform the client about this fact.

A.5. UC5 - Immutable BGP AS number and peer type

An autonomous system (AS) number is assigned and used primarily with BGP to uniquely identify each network system. Changing AS attribute will cause it to delete all the current routing entries and learning new ones, during which process it might lead to traffic disruption. It is usually not allowed to modify the AS attribute once it is configured unless all BGP configurations are removed.

Another example is the type attribute of BGP neighbors. The peer type of the BGP neighbor is closely related to the network topology: external BGP (EBGP) peer type relationships are established between BGP routers running in different ASs; while internal BGP (IBGP) peer type relationships are established between BGP routers running in the same AS. Thus BGP peer type cannot be changed to the value which does not match the actual one. Since there are EBGP/IBGP-specific configurations which need to reference the "peer-type" node (e.g., in "when" statement) and be conditionally available, it can only be modelled as "config true" but immutable.

Following is the fragment of a simplified BGP module with the /bgp/as and /bgp/neighbor/peer-type defined as immutable:

```
   container bgp {
     leaf as {
       im:immutable "create delete";
       type inet:as-number;
       mandatory true;
       description
         "Local autonomous system number of the router.";
     }
     list neighbor {
       key "remote-address";
       leaf remote-address {
         type inet:ip-address;
         description
           "The remote IP address of this entry's BGP peer.";
       }
       leaf peer-type {
         im:immutable "create delete";
         type enumeration {
           enum ebgp {
            description
              "External (EBGP) peer.";
           }
           enum ibgp {
             description
               "Internal (IBGP) peer.";
           }
         }
         mandatory true;
         description
           "Specify the type of peering session associated with this
            neighbor. The value can be IBGP or EBGP.";
       }
       leaf ebgp-max-hop {
         when "../peer-type='ebgp'";
         type uint32 {
           range "1..255";
         }
         description
           "The maximum number of hops when establishing an EBGP peer
            relationship with a peer on an indirectly-connected network.
            By default, an EBGP connection can be set up only on a
            directly-connected physical link.";
       }
     }
   }
```

A.6. UC6 - Modeling existing data handling behavior in other standard organizations

A number of standard organizations and industry groups (ITU-T, 3GPP, ORAN) already use concepts similar to immutability. These modeling concepts sometimes go back to more than 10 years and cannot be and will not be changed irrespective of the YANG RFCs. Some of these organizations are introducing YANG modelling. Without a formal YANG statement to define data nodes immutable the property is only defined in plain Englist text in the description statement. The immutable extension and/or metadata annotation can be used to define these existing model properties in a machine-readable way.

Appendix B. Existing implementations

There are already a number of full or partial implementations of immutability.

3GPP TS 32.156 [TS32.156] and 28.623 [TS28.623]: Requirements and a partial solution

ITU-T using ONF TR-531[TR-531] concept on information model level but no YANG representation.

Ericsson: requirements and solution

YumaPro: requirements and solution

Nokia: partial requirements and solution

Huawei: partial requirements and solution

Cisco using the concept at least in some YANG modules

Junos OS provides a hidden and immutable configuration group called junos-defaults

Appendix C. Changes between revisions

Note to RFC Editor (To be removed by RFC Editor)

v04 - v05

* Emphasized that the proposal tries to formally document existing allowed behavior

* Reword the abstract and introduction sections;

* Restructure the document;

* Simplified the interface example in Appendix;

* Add immutable BGP AS number and peer-type configuration example.

* Added temporary section in Annex B about list of existing non-standard solutions

* Clarified inheritance of immutability

* Clarified that this draft is not dependent on the existence of the <system> datastore.

v03 - v04

* Clarify how immutable flag interacts with NACM mechanism.

v02 - v03

* rephrase and avoid using "server MUST reject" statement, and try to clarify that this documents aims to provide visibility into existing immutable behavior;

* Add a new section to discuss the inheritance of immutability;

* Clarify that deletion to an immutable node in <running> which is instantiated in <system> and copied into <running> should always be allowed;

* Clarify that write access restriction due to general YANG rules has no need to be marked as immutable.

* Add an new section named "Acknowledgements";

* editoral changes.

v01 - v02

* clarify the relation between the creation/deletion of the immutable data node with its parent data node;

* Add a "TODO" comment about the inheritance of the immutable property;

* Define that the server should reject write attempt to the immutable data node at an <edit-config> operation time, rather than waiting until a <commit> or <validate> operation takes place;

v00 - v01

* Added immutable extension

* Added new use-cases for immutable extension and annotation

* Added requirement that an update that means no effective change should always be allowed

* Added clarification that immutable is only applied to read-write datastore

* Narrowed the applied scope of metadata annotation to list/leaf-list instances

Appendix D. Open Issues tracking

* Can we do better about the "immutable" terminology?

* Is a Boolean type for immutable metadata annotation sufficient?

* Can immutable data be removed due to a when or choice statement?

Authors' Addresses

Qiufang Ma
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: maqiufang1@huawei.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: bill.wu@huawei.com

Balazs Lengyel
Ericsson
Email: balazs.lengyel@ericsson.com

Hongwei Li
HPE
Email: flycoolman@gmail.com

IETF
Internet-Draft
Intended status: Standards Track
Expires: 9 September 2023

K. Moriarty
Center for Internet Security (CIS)

8 March 2023

Security Considerations Template for YANG Module Documents
draft-moriarty-yangsecuritytext-02

Abstract

This document includes the template text agreed upon by the Operations Area and Security Area for inclusion in YANG documents. The best practices are updated as needed and will result in updates to this template for use to provide a consistent set of security considerations for authors, developers, and implementors.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document includes the template text agreed upon by the Operations Area and Security Area for inclusion in YANG documents. The best practices are updated as needed and will result in updates to this template for use to provide a consistent set of security considerations for authors, developers, and implementors.

Updates may be made through errata or a publication of an updated document for ease of use by the IETF and other standards organizations. The current version is maintained on a wiki.

2. YANG Security Considerations Template

The following template text, in addition to the guidance provided by the Security Area Directors in the Security Area wiki, must be included in the applicable IETF YANG publications. The text is provided as a template for use by other organizations with a requirement to reference it appropriately to this document.

This RFC contains text intended for use as a template as designated below by the markers

<BEGIN TEMPLATE TEXT> and <END TEMPLATE TEXT>

or other clear designation. Such Template Text is subject to the provisions of Section 9(b) of the

<BEGIN TEMPLATE TEXT>

Security Considerations

The YANG module(s) 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 [RFC 8446].

The Network Configuration Access Control Model (NACM) [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.

-- if you have any writable data nodes (those are all the -- "config true" nodes, and remember, that is the default) -- describe their specific sensitivity or vulnerability.

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:

<list subtrees and data nodes and state why they are sensitive>

-- for all YANG modules you must evaluate whether any readable data -- nodes (those are all the "config false" nodes, but also all other -- nodes, because they can also be read via operations like get or -- get-config) are sensitive or vulnerable (for instance, if they -- might reveal customer information or violate personal privacy -- laws such as those of the European Union if exposed to -- unauthorized parties)

Some of the readable data nodes in this YANG module 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. These are the subtrees and data nodes and their sensitivity/vulnerability:

<list subtrees and data nodes and state why they are sensitive>

-- if your YANG module has defined any rpc operations -- describe their specific sensitivity or vulnerability.

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

<list RPC operations and state why they are sensitive>

<END TEMPLATE TEXT>

Note: [RFC 8446], [RFC6241], [RFC6242], [RFC8341], and [RFC8040] must be "normative references".

3. Security Considerations

This document defines a template to provide consistent YANG Security Considerations on publications by the IETF and other standards bodies and organizations.

4. IANA Considerations

This memo includes no request to IANA.

5. Contributors

Thank you to reviewers and contributors who helped to improve the security consdierations for YANG. The text has been developed and refined over many years on an Operations Area working group mailing list and to a Security Area wiki. Revisions have been made by IETF Security Area Directors and Operations Area Directors similar to the template for SNMP security considerations. Thank you to the following known contributors: Sean Turner, Stephen Farrell, Beniot Claise, and Erik Rescorla.

6. References

6.1. 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>.

[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>.

6.2. Informative References

[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>.

[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>.

[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>.

[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>.

Appendix A. Change Log

Note to RFC Editor: if this document does not obsolete an existing RFC, please remove this appendix before publication as an RFC.

Author's Address

Kathleen M. Moriarty
Center for Internet Security (CIS)
31 Tech Valley Drive
East Greenbush, NY,
United States of America
Email: Kathleen.Moriarty.ietf@gmail.com