Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

A. Clemm
Huawei - Futurewei Technologies, Inc.
E. Voit
Cisco Systems
X. Liu
Volta Networks
I. Bryskin
T. Zhou
G. Zheng
Huawei
H. Birkholz
Fraunhofer SIT

October 22, 2018

Smart Filters for Push Updates
draft-clemm-netmod-push-smart-filters-01

Abstract

This document defines a YANG model for Smart Filters for push updates. Smart Filters allow to filter push updates based on values of pushed datastore nodes and/or state, such as previous updates. Smart Filters provide an important building block for service assurance and network automation.

This revision of the document is intended as a placeholder, containing the problem statement of draft-clemm-netconf-push-smart-filters-ps-00 that has recently expired. The YANG model itself still needs to be defined.

Table of Contents

1. Introduction

YANG-Push [yang-push] allows client applications to subscribe to continuous datastore updates without needing to poll. YANG-Push subscriptions allow client applications to select which datastore nodes are of interest. For this purpose, filters that act as node selectors are offered. However, what is currently not supported are filters that filter updates based on values, such as sending updates only when the value falls within a certain range. Also not supported are filters that would require additional state, such as sending updates only when the value exceeds a certain threshold for the first time but not again until the threshold is cleared. We refer to such filters as "Smart Filters", with further subcategories of "smart stateless filters" and "smart stateful filters", respectively.

Smart Filters involve more complex subscription and implementation semantics than the simple selection filters that are currently offered as part of YANG-Push. They involve post processing of updates that goes beyond basic update generation for polling avoidance and place additional intelligence at the server. Because of this, Smart Filter functionality was not included in the YANG-Push

specification, although it was recognized that YANG-Push could be extended to include such functionality if needed. This is the purpose of this specification.

Smart Filters facilitate service assurance, because they allow client applications to focus on "outliers" and updates that signify exceptions and conditions of interest have the biggest operational significance. They save network resources by avoiding the need to stream updates that would be discarded anyway, and allow applications to scale better since larger networks imply a larger amount of Smart Filtering operations delegated away from the application to the network. Smart Filters also facilitate network automation as they constitute an important ingredient to specify triggers for automated actions.

2. Key Words

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

3. Definitions and Acronyms

Datastore node: An instance of management information in a datastore. Also known as "object".

Smart Filter: A filter that involves some processing, such as comparing values or differentiating behavior depending on state.

TCA: Threshold Crossing Alert.

YANG-Push: A server capability that allows client applications to subscribe to network management datastore updates.

4. Problem Statement

YANG-Push provides client applications with the ability to subscribe to continuous updates from network management datastores, obviating the need to perform polling and resulting in more robust and efficient applications. However, many applications do not require every update, only updates that are of certain interest.

For example, an update concerning interface utilization may be only needed when a certain utilization level is breached. Sending continuous updates when utilization is low might divert processing resources away from updates regarding interfaces whose utilization

level may reach a critical point that requires attention. Doing so will require a filter based on an object value. Even sending continuous updates when utilization is high may be too much and counterproductive. It may be sufficient to send an update when a threshold is breached to raise a flag of attention, but then not to continue sending updates while the condition still persists but simply let the client application know when the threshold is cleared. This behavior cannot be accomplished simply by a value-based filter, but requires additional state to be maintained (so that the server has a memory whether or not the condition of a breached threshold has already been reported in prior update cycles).

What is needed are "Smart Filters" that provide the ability to apply filters based on object values, possibly also state state. Smart Filters are useful for Service Assurance applications that need to monitor operational data for values that fall outside normal operational ranges. They are also useful for network automation, in which automated actions are automatically triggered based on when certain events in the network occur while certain conditions hold. A YANG-Push subscription with a Smart Filter can in effect act as a source for such events. Combined with an optional check for a condition when an event is observed, this can serve as the basis of action triggers.

Smart Filters for Push Updates will provide support for the following features:

- Support for Smart Filter extensions to YANG-Push subscriptions. The targeted model takes a "base" YANG-Push subscription and subjects updates to an additional filtering stage that is based on value.

- Support for selected stateful filters:

  * This includes specifically support for generalized "threshold crossing alert" filters, or filters that provide an update only when a datastore node's value passes a filter for the first time, and not again until the datastore node's value passes a counter filter. In effect, the support involves attaching filter and counter filter to a datastore node, including a switch at the datastore node indicating which filter is in effect, and providing a distinction in the update which filter (e.g. onset of clear) was applied.

  * It may include additional filters, such a "recent high water mark" filters that allow to specify a time horizon until the current high water mark clears. A recent high water mark

filter sends an update to an object only if its new value is greater than the last value that had been previously reported.

- In addition to new filters, support for features to make them easier to use:
  - Support for refined on-change update semantics that allow client to distinguish whether datastore node values were omitted or included because the datastore node was created or deleted, or because the datastore node's value fell outside filter range.
  - Support for a heartbeat that indicates that a filter is still in effect after a longer period of inactivity.

It is easy to conceive of filters that are very smart and powerful yet also very complex. While filters as defined in YANG-Push may be a tad too simple for the applications envisioned here, it is important to keep filters still simple enough to ensure broad implementation and support by networking devices. The purpose of Smart Filters defined in this effort is to address the 90% of cases that can be addressed using 10% of the complexity. Items like the following will therefore be outside the scope:

- Filters that involve freely programmable logic.
- Filters that aggregate or otherwise process information over time. An example would be filters that compute an aggregate over a time series of data (e.g. a datastore node's average or top percentile value)
- Filters that aggregate or compare values of several datastore nodes (e.g. the maximum or average from datastore nodes in a list).

5. Smart Filter Data Model

The following section contains an initial YANG data model for smart filters. The model is at this point still incomplete and included as a starting point only. At this point, the model defines a simple threshold filter. When used with a subscription, objects that meet the filter criterion (i.e. the threshold comparison) are included in the update whereas any other object is filtered.

The model will be extended to define a full "smart threshold" model in a later revision. This will add the feature of a hysteresis threshold, i.e. a counter threshold that allows to define when a crossed threshold should be cleared. The value of the hysteresis

threshold can be set to a lower value than the threshold itself to avoid unnecessary updates in case of oscillations). It will also add a notion of state to remember whether a threshold crossing has already been reported, to avoid repeated inclusion of objects in updates that remain above their threshold. By including metadata, clients will be able to distinguish between the violation and the clearing of thresholds.

The model will furthermore be extended for smart filters that are not threshold-related, such as the previously mentioned recent high water marks.

```
<CODE BEGINS> file "ietf-smart-filter@2018-10-22.yang"
module ietf-smart-filter {
  yang-version "1.1";
  namespace "urn:ietf:params:xml:ns:yang:ietf-smart-filter";
  prefix "sf";

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

  import ietf-subscribed-notifications {

    prefix sn;
    reference
      "draft-ietf-netconf-subscribed-notifications:
       Customized Subscriptions to a Publisher's Event Streams

       NOTE TO RFC Editor: Please replace above reference to
       draft-ietf-netconf-subscribed-notifications with RFC number
       when published (i.e. RFC xxxx).";
  }

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

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

     Editor:   Eric Voit
               <mailto:evoit@cisco.com>

     Editor:   Xufeng Liu
```

```
                <mailto:xufeng.liu.ietf@gmail.com>

       Editor:  Igor Bryskin
                <mailto:igor.bryskin@huawei.com>

       Editor:  Tianran Zhou
                <mailto:zhoutianran@huawei.com>

       Editor:  Guangying Zheng
                <mailto:zhengguangying@huawei.com>

       Editor:  Henk Birkholz
                <mailto:henk.birkholz@sit.fraunhofer.de>";

   description
     "This module contains YANG specifications for smart filter.

     Copyright (c) 2018 IETF Trust and the persons identified as authors
     of the code.  All rights reserved.

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

         This version of this YANG module is part of
         draft-clemm-netmod-push-smart-filters-01; see the RFC itself
                 for full legal notices.

         NOTE TO RFC EDITOR: Please replace above reference to
         with RFC number when
         published (i.e. RFC xxxx).";

   revision 2018-10-22 {
     description
       "Initial revision.
       NOTE TO RFC EDITOR:
       (1)Please replace the above revision date to
       the date of RFC publication when published.
       (2) Please replace the date in the file name
       (ietf-smart-filter@2018-10-22.yang) to the date of RFC
       publication.
       (3) Please replace the following reference to
       draft-clemm-netmod-push-smart-filters-01 with RFC number when
       published (i.e. RFC xxxx).";
     reference
       "draft-clemm-netmod-push-smart-filters-01";
```

```
   }

  /*
   * IDENTITIES
   */

   /* Smart-filter type identities  */

   identity smart-filter {
      description
       "A base identity that represents the smart filter types. ";
   }

   identity smart-filter-threshold {
      base smart-filter;
      description
       "An identity instance based on smart-filter, which support
        filter the push data by fix threshold value.";
   }

  /*
   * TYPE DEFINITIONS
   */
   typedef sf-op-type {
     type enumeration {
       enum eq {
         description "equal to";
       }
       enum gt {
         description "greater than";
       }
       enum ge {
         description "greater than or equal to";
       }
       enum lt {
         description "less than";
       }
       enum le {
         description "less than or equal to";
       }
     }
     description "A boolean comparator for an object and a data value.
              Include: eq, gt, ge, lt, le.";
   }

  /*
   * GROUP DEFINITIONS
   */
```

```
   grouping sf-threshold{
     description
       "the group for threshold filter";
     leaf filter-node {
       if-feature "sn:xpath";
       type yang:xpath1.0;
       description
         "This parameter contains an XPath expression identifying
          the node of the target filter.";
       }

     leaf threshold-value {
       type string;
       description "threshold value";
     }

     leaf op-type {
       type sf-op-type;
       description "comparison operator";
     }
   }

   //augment statements
   augment "/sn:subscriptions/sn:subscription" {
     description "add the smart filter container";
     container smart-filter {
       description "It concludes filter configurations";

       choice filter-type {
         description
           "Select different smart filter";
         case threshold-filter {
           description
             "threshold-filter";
           uses sf-threshold;
         }
       }
     }
   }
 }
 <CODE ENDS>
```

6. IANA Considerations

RFC Ed.: In this section, replace all occurrences of 'XXXX' with the actual RFC number (and remove this note).

IANA is requested to assign a new URI from the IETF XML Registry [RFC3688]. The following URI is suggested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-smart-filter
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.
```

This document also requests a new YANG module name in the YANG Module Names registry [RFC7950] with the following suggestion:

```
name: ietf-ioam
namespace: urn:ietf:params:xml:ns:yang:ietf-smart-filter
prefix: sf
reference: RFC XXXX
```

7. Security Considerations

The application of Smart Filters requires a certain amount of processing resources at the server. An attacker could attempt to attack a server by creating YANG-push subscriptions with a large number of complex Smart Filters in an attempt to diminish server resources. Server implementations can guard against such scenarios in several ways. For one, they can implement NACM in order to require proper authorization for requests to be made. Second, server implementations can reject requests made for a a larger number of Smart Filters than the implementation can reasonably sustain.

8. Normative References

[notif-sub] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Custom Subscriptions to a Publisher's Event Streams", June 2018, <https://datatracker.ietf.org/doc/draft-ietf-netconf-subscribed-notifications/>.

[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>.

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

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

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

[yang-push] Clemm, A., Voit, E., Gonzalez Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscribing to YANG datastore push updates", July 2018, <https://datatracker.ietf.org/doc/draft-ietf-netconf-yang-push/>.

Authors' Addresses

Alexander Clemm
Huawei - Futurewei Technologies, Inc.
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: ludwig@clemm.org

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Xufeng Liu
Volta Networks

Email: xufeng.liu.ietf@gmail.com

Igor Bryskin
Huawei

Email: igor.bryskin@huawei.com

Tianran Zhou
Huawei

Email: zhoutianran@huawei.com

Guangying Zheng
Huawei

Email: zhengguangying@huawei.com

Henk Birkholz
Fraunhofer SIT

Email: henk.birkholz@sit.fraunhofer.de

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

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

July 2, 2018

Common Interface Extension YANG Data Models
draft-ietf-netmod-intf-ext-yang-06

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. These properties are common to many types of interfaces on network routers and switches and are implemented by multiple network equipment vendors with similar semantics, even though some of the features are not formally defined in any published standard.

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 draft is to provide a standard namespace and path for these configuration items regardless of the underlying interface type. For example a standard namespace and path 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 draft is to define common paths and leaves for the configuration of these equivalent features in a uniform way, making it easier for users of the YANG model to access these features in a vendor independent way. Where necessary, a description of the expected behavior is also provided with the aim of ensuring vendors implementations are consistent with the specified behaviour.

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

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

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

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

1.1. Terminology

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

1.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration (read-write), and "ro" means state data (read-only).
- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list or leaf-list.
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Objectives

The aim of the YANG modules contained in this draft is to provide standard definitions for common interface based configuration on network devices.

The expectation is that the YANG leaves that are being defined are fairly widely implemented by network vendors. However, the features described here are mostly not backed by formal standards because they are fairly basic in their behavior and do not need to interoperate with other devices. Where required a concise explanation of the expected behavior is also provided to ensure consistency between vendors.

3. Interfaces Common Module

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

The module provides:

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

The "ietf-interfaces-common" YANG module has the following structure:

```
module: ietf-interfaces-common
  augment /if:interfaces/if:interface:
    +--rw carrier-delay {carrier-delay}?
    |  +--rw down?                  uint32
    |  +--rw up?                    uint32
    |  +--ro carrier-transitions?   yang:counter64
    |  +--ro timer-running?         enumeration
    +--rw dampening! {dampening}?
    |  +--rw half-life?           uint32
    |  +--rw reuse?               uint32
    |  +--rw suppress?            uint32
    |  +--rw max-suppress-time?   uint32
    |  +--ro penalty?             uint32
    |  +--ro suppressed?          boolean
    |  +--ro time-remaining?      uint32
    +--rw encapsulation
    |  +--rw (encaps-type)?
    +--rw loopback?          identityref {loopback}?
    +--rw l2-mtu?            uint16 {configurable-l2-mtu}?
    +--rw forwarding-mode?   identityref {forwarding-mode}?
  augment /if:interfaces/if:interface:
    +--rw parent-interface    if:interface-ref {sub-interfaces}?
```

3.1. Carrier Delay

The carrier delay feature augments the IETF interfaces data model with configuration for a simple algorithm that is used, generally on physical interfaces, to suppress short transient changes in the interface link state. It can be used in conjunction with the dampening feature described in Section 3.2 to provide effective control of unstable links and unwanted state transitions.

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

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

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

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

3.2. 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 nominally 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. Implementations are not required to use a penalty of 1000 units in their dampening algorithm, but should ensure that the Suppress Threshold and Reuse Threshold values are scaled relative to the nominal 1000 unit penalty to ensure that the same configuration

values provide consistent behaviour. The configurable values are described in more detail below.

3.2.1. 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 the default or configured suppress threshold, the interface is placed in a dampened state.

3.2.2. Half-Life Period

The half-life period determines how fast the accumulated penalties can decay exponentially. Any penalties that have been accumulated on a flapping interface are reduced by half after each half-life period.

3.2.3. 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 dampened state and allowed to go up.

3.2.4. Maximum Suppress Time

The maximum suppress time represents the maximum amount of time an interface can remain dampened when a 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.

3.3. Encapsulation

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

The different encapsulations themselves are defined in separate YANG modules defined in other documents that augument the encapsulation choice statement. For example the Ethernet specific basic 'dot1q-vlan' encapsulation is defined in ietf-if-l3-vlan.yang and the

'flexible' encapsulation is defined in ietf-flexible-encapsulation.yang, both modules from [I-D.ietf-netmod-sub-intf-vlan-model].

3.4. 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.

3.5. Layer 2 MTU

A layer 2 MTU configuration leaf (l2-mtu) is provided to specify the maximum size of a layer 2 frame that may be transmitted or received on an interface. The layer 2 MTU includes the overhead of the layer 2 header and the maximum length of the payload, but excludes any frame check sequence (FCS) bytes. The payload MTU available to higher layer protocols is calculated from the l2-mtu leaf after taking the layer 2 header size into account.

For Ethernet interfaces carrying 802.1Q VLAN tagged frames, the l2-mtu excludes the 4-8 byte overhead of any known (e.g. explicitly matched by a child sub-interface) 801.1Q VLAN tags.

3.6. 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.

3.7. 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.

YANG Modules can conditionally use this leaf as a simple mechanism to determine whether particular types of configuration are valid. YANG modules can write 'must' statements to check whether the forwarding mode leaf has been configured, and if it is, then validate that the specified configuration is consistent with any forwarding mode that has also been configured. E.g., a layer 2 QoS policy YANG module could ensure that it is only applied to a interface forwarding traffic at layer 2 by checking whether the forwarding-mode leaf exists, and if it does then also ensure that it has been set to 'layer-2-forwarding'.

The following forwarding modes are defined:

- Optical Layer - Traffic is being forwarded at the optical layer. This includes DWDM or OTN based switching.

- Layer 2 - Layer 2 based forwarding, such as Ethernet/VLAN based switching, or L2VPN services.

- Network Layer - Network layer based forwarding, such as IP, MPLS, or L3VPNs.

4. 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-interfaces-ethernet-like" YANG module has the following structure:

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

5. Interfaces Common YANG Module

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

```
<CODE BEGINS> file "ietf-interfaces-common@2018-07-02.yang"
module ietf-interfaces-common {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-interfaces-common";

  prefix if-cmn;

  import ietf-yang-types {
    prefix yang;
  }
```

```
   import ietf-interfaces {
     prefix if;
   }

   import iana-if-type {
     prefix ianaift;
   }

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

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

      WG Chair: Lou Berger
                <mailto:lberger@labn.net>

      WG Chair: Joel Jaeggli
                <mailto:joelja@gmail.com>

      WG Chair: Kent Watsen
                <mailto:kwatsen@juniper.net>

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

   description
     "This module contains common definitions for extending the IETF
      interface YANG model (RFC 7223) with common configurable layer 2
      properties.

      Copyright (c) 2016-2018 IETF Trust and the persons identified
      as authors of the code.  All rights reserved.

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

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

   revision 2018-07-02 {
     description
       "Initial version";
```

```
    reference "Internet draft: draft-ietf-netmod-intf-ext-yang-06";
  }

  feature carrier-delay {
    description
      "This feature indicates that configurable interface
       carrier delay is supported, which is a feature is used to
       limit the propagation of very short interface link state
       flaps.";
    reference "RFC XXX, Section 3.1 Carrier Delay";
  }

  feature dampening {
    description
      "This feature indicates that the device supports interface
       dampening, which is a feature that is used to limit the
       propagation of interface link state flaps over longer
       periods";
    reference "RFC XXX, Section 3.2 Dampening";
  }

  feature loopback {
    description
      "This feature indicates that configurable interface loopback
       is supported.";
    reference "RFC XXX, Section 3.4 Loopback";
  }

  feature configurable-l2-mtu {
    description
      "This feature indicates that the device supports configuring
       layer 2 MTUs on interfaces.  Such MTU configurations include
       the layer 2 header overheads (but exclude any FCS overhead).
       The payload MTU available to higher layer protocols is either
       derived from the layer 2 MTU, taking into account the size of
       the layer 2 header, or is further restricted by explicit layer
       3 or protocol specific MTU configuration.";
    reference "RFC XXX, Section 3.5 Layer 2 MTU";
  }

  feature sub-interfaces {
    description
      "This feature indicates that the device supports the
       instantiation of sub-interfaces.  Sub-interfaces are defined
       as logical child interfaces that allow features and forwarding
       decisions to be applied to a subset of the traffic processed
       on the specified parent interface.";
    reference "RFC XXX, Section 3.6 Sub-interface";
```

```
   }

   feature forwarding-mode {
     description
       "This feature indicates that the device supports the
        configurable forwarding mode leaf";
     reference "RFC XXX, Section 3.7 Forwarding Mode";
   }

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

        New or custom interface types can derive from this type to
        inherit generic sub-interface configuration";
     reference "RFC XXX, Section 3.6 Sub-interface";
   }

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

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

   identity loopback {
     description "Base identity for interface loopback options";
     reference "RFC XXX, section 3.4";
   }
   identity loopback-internal {
     base loopback;
     description
       "All egress traffic on the interface is internally looped back
        within the interface to be received on the ingress path.";
     reference "RFC XXX, section 3.4";
   }
   identity loopback-line {
     base loopback;
     description
       "All ingress traffic received on the interface is internally
        looped back within the interface to the egress path.";
```

```
    reference "RFC XXX, section 3.4";
  }
  identity loopback-connector {
    base loopback;
    description
      "The interface has a physical loopback connector attached
       that loops all egress traffic back into the interface's
       ingress path, with equivalent semantics to loopback-internal";
    reference "RFC XXX, section 3.4";
  }

  identity forwarding-mode {
    description "Base identity for forwarding-mode options.";
    reference "RFC XXX, section 3.7";
  }
  identity optical-layer {
    base forwarding-mode;
    description
      "Traffic is being forwarded at the optical layer.  This
       includes DWDM or OTN based switching.";
    reference "RFC XXX, section 3.7";
  }
  identity layer-2-forwarding {
    base forwarding-mode;
    description
      "Layer 2 based forwarding, such as Ethernet/VLAN based
       switching, or L2VPN services.";
    reference "RFC XXX, section 3.7";
  }
  identity network-layer {
    base forwarding-mode;
    description
      "Network layer based forwarding, such as IP, MPLS, or L3VPNs.";
    reference "RFC XXX, section 3.7";
  }


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

    /*
```

```
      * Defines standard YANG for the Carrier Delay feature.
      */
     container carrier-delay {
       if-feature "carrier-delay";
       description
         "Holds carrier delay related feature configuration";
       leaf down {
         type uint32;
         units milliseconds;
         description
           "Delays the propagation of a 'loss of carrier signal' event
            that would cause the interface state to go down, i.e. the
            command allows short link flaps to be suppressed. The
            configured value indicates the minimum time interval (in
            milliseconds) that the carrier signal must be continuously
            down before the interface state is brought down. If not
            configured, the behaviour on loss of carrier signal is
            vendor/interface specific, but with the general
            expectation that there should be little or no delay.";
       }
       leaf up {
         type uint32;
         units milliseconds;
         description
           "Defines the minimum time interval (in milliseconds) that
            the carrier signal must be continuously present and error
            free before the interface state is allowed to transition
            from down to up.  If not configured, the behaviour is
            vendor/interface specific, but with the general
            expectation that sufficient default delay should be used
            to ensure that the interface is stable when enabled before
            being reported as being up.  Configured values that are
            too low for the hardware capabilties may be rejected.";
       }
       leaf carrier-transitions {
         type yang:counter64;
         units transitions;
         config false;
         description
           "Defines the number of times the underlying carrier state
            has changed to, or from, state up.  This counter should be
            incremented even if the high layer interface state changes
            are being suppressed by a running carrier-delay timer.";
       }
       leaf timer-running {
         type enumeration {
           enum none {
             description
```

```
              "No carrier delay timer is running.";
          }
          enum up {
            description
              "Carrier-delay up timer is running.  The underlying
               carrier state is up, but interface state is not
               reported as up.";
          }
          enum down {
            description
              "Carrier-delay down timer is running.  Interface state
               is reported as up, but the underlying carrier state is
               actually down.";
          }
        }
        default "none";
        config false;
        description
          "Reports whether a carrier delay timer is actively running,
           in which case the interface state does not match the
           underlying carrier state.";
      }

      reference "RFC XXX, Section 3.1 Carrier Delay";
    }

    /*
     * Augments the IETF interfaces model with a container to hold
     * generic interface dampening
     */
    container dampening {
      if-feature "dampening";
      presence
        "Enable interface link flap dampening with default settings
         (that are vendor/device specific)";
      description
        "Interface dampening limits the propagation of interface link
         state flaps over longer periods";
      reference "RFC XXX, Section 3.2 Dampening";
      leaf half-life {
        type uint32;
        units seconds;
        description
          "The Time (in seconds) after which a penalty reaches half
           its original value. Once the interface has been assigned
           a penalty, the penalty is decreased by half after the
           half-life period. For some devices, the allowed values may
           be restricted to particular multiples of seconds. The
```

```
           default value is vendor/device specific.";
        reference "RFC XXX, Section 3.3.2 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 nominally 1000 units.";
        reference "RFC XXX, Section 3.2.3 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 nominally 1000 units.";
        reference "RFC XXX, Section 3.2.1 Suppress Threshold";
      }

      leaf max-suppress-time {
        type uint32;
        units seconds;
        description
          "Maximum time (in seconds) that an interface can be
           suppressed. This value effectively acts as a ceiling that
           the penalty value cannot exceed.  The default value is
           vendor/device specific.";
        reference "RFC XXX, Section 3.2.4 Maximum Suppress Time";
      }

      leaf penalty {
        type uint32;
        config false;
        description
          "The current penalty value for this interface.  When the
           penalty value exceeds the 'suppress' leaf then the
           interface is suppressed (i.e. held down).";
        reference "RFC XXX, Section 3.2 Dampening";
      }

      leaf suppressed {
        type boolean;
        default "false";
```

```
        config false;
        description
          "Represents whether the interface is suppressed (i.e. held
           down) because the 'penalty' leaf value exceeds the
           'suppress' leaf.";
        reference "RFC XXX, Section 3.2 Dampening";
      }

      leaf time-remaining {
        when '../suppressed = "true"' {
          description
            "Only suppressed interfaces should have a time remaining.";
        }
        type uint32;
        units seconds;
        config false;
        description
          "For a suppressed interface, this leaf represents how long
           (in seconds) that the interface will remain suppressed
           before it is allowed to go back up again.";
        reference "RFC XXX, Section 3.2 Dampening";
      }
    }

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

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

```
      description
        "Holds the OSI layer 2 encapsulation associated with an
         interface";
      choice encaps-type {
        description
          "Extensible choice of layer 2 encapsulations";
        reference "RFC XXX, Section 3.3 Encapsulation";
      }
    }

     /*
      * Various types of interfaces support loopback configuration,
      * any that are supported by YANG should be listed here.
      */
    leaf loopback {
      when "derived-from-or-self(../if:type,
                                 'ianaift:ethernetCsmacd') or
            derived-from-or-self(../if:type, 'ianaift:sonet') or
            derived-from-or-self(../if:type, 'ianaift:atm') or
            derived-from-or-self(../if:type, 'ianaift:otnOtu')" {
        description
          "All interface types that support loopback configuration.";
      }
      if-feature "loopback";
      type identityref {
        base loopback;
      }
      description "Enables traffic loopback.";
      reference "RFC XXX, Section 3.4 Loopback";
    }

    /*
     * Many types of interfaces support a configurable layer 2 MTU.
     */
    leaf l2-mtu {
      if-feature "configurable-l2-mtu";
      type uint16 {
        range "64 .. 65535";
      }
      description
        "The maximum size of layer 2 frames that may be transmitted
         or received on the interface (excluding any FCS overhead).
         In the case of Ethernet interfaces it also excludes the
         4-8 byte overhead of any known (i.e. explicitly matched by
         a child sub-interface) 801.1Q VLAN tags.";
      reference "RFC XXX, Section 3.5 Layer 2 MTU";
    }
```

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

       description
         "The forwarding mode that the interface is operating in.";
       reference "RFC XXX, Section 3.7 Forwarding Mode";
     }
   }

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

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

       type if:interface-ref;

       mandatory true;
       description
         "This is the reference to the parent interface of this
          sub-interface.";
       reference "RFC XXX, Section 3.6 Sub-interface";
     }
   }
 }
```

<CODE ENDS>

6. Interfaces Ethernet-Like YANG Module

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

```
<CODE BEGINS> file "ietf-interfaces-ethernet-like@2017-07-03.yang"
module ietf-interfaces-ethernet-like {
  yang-version 1.1;

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

  prefix ethlike;

  import ietf-interfaces {
    prefix if;
  }

  import ietf-yang-types {
    prefix yang;
  }

  import iana-if-type {
    prefix ianaift;
  }

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

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

     WG Chair: Lou Berger
               <mailto:lberger@labn.net>

     WG Chair: Kent Watsen
               <mailto:kwatsen@juniper.net>

     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.

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

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

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

  revision 2017-07-03 {
    description "Updated to conform to NMDA architecture";

    reference
      "Internet draft: draft-ietf-netmod-intf-ext-yang-05";
  }

  /*
   * 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:l2vlan') 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 {
        type yang:mac-address;
        description
```

```
            "The MAC address of the interface.";
          }

          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.";
          }

          container statistics {
            config false;
            description
              "Packet statistics that apply to all Ethernet-like
               interfaces";
            leaf in-drop-unknown-dest-mac-pkts {
              type yang:counter64;
              units frames;
              description
                "A count of the number of frames that were well formed,
                 but otherwise dropped because the destination MAC
                 address did not pass any ingress destination MAC address
                 filter.

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

                 Discontinuities in the values of this counters in this
                 container 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>
```

7. 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].

7.1. Carrier delay configuration

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

```
<?xml version="1.0" encoding="utf-8"?>
<interfaces
 xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
 xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
xmlns:if-cmn="urn:ietf:params:xml:ns:yang:ietf-interfaces-common">
  <interface>
    <name>eth0</name>
    <type>ianaift:ethernetCsmacd</type>
    <if-cmn:carrier-delay>
      <if-cmn:down>0</if-cmn:down>
      <if-cmn:up>1000</if-cmn:up>
    </if-cmn:carrier-delay>
  </interface>
</interfaces>
```

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

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

7.2. 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>
    <dampening
     xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces-common">
      <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>
```

7.3. 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>
      <ethernet-like
  xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces-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 an explicit MAC address being configured on interface eth0.

```
<?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-interfaces-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>
    <ethernet-like
  xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces-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>
```

8. Acknowledgements

The authors wish to thank Eric Gray, Ing-Wher Chen, Juergen Schoenwaelder, Ladislav Lhotka, Mahesh Jethanandani, Michael Zitao, Neil Ketley, Qin Wu, William Lupton, Xufeng Liu, and Andy Bierman for their helpful comments contributing to this draft.

9. ChangeLog

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

9.1. Version -06

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

9.2. Version -05

- Incorporate feedback from Andy Bierman

9.3. Version -04

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

9.4. Version -03

- Fixed incorrect module name references, and updated tree output

9.5. Version -02

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

10. IANA Considerations

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

11. Security Considerations

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

There are a number of data nodes defined in this YANG module which are writable/creatable/deletable (i.e. config true, which is the

default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g. edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

11.1. interfaces-common.yang

The interfaces-common 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:

- /if:interfaces/if:interface/loopback

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

- /if:interfaces/if:interface/carrier-delay/down
- /if:interfaces/if:interface/carrier-delay/up
- /if:interfaces/if:interface/dampening/half-life
- /if:interfaces/if:interface/dampening/reuse
- /if:interfaces/if:interface/dampening/suppress
- /if:interfaces/if:interface/dampening/max-suppress-time

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

- /if:interfaces/if:interface/encapsulation
- /if:interfaces/if:interface/l2-mtu
- /if:interfaces/if:interface/forwarding-mode

Normally devices will not allow the parent-interface leaf to be changed after the interfce has been created. If an implementation

did allow the parent-interface leaf to be changed then it could cause all traffic on the affected interface to be dropped. The affected leaf is:

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

11.2. interfaces-ethernet-like.yang

Generally, the configuration nodes in the interfaces-ethernet-like YANG module are concerned with configuration that is common across all types of Ethernet-like interfaces. Currently, the module 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

12. References

12.1. Normative References

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

[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>.

12.2. Informative References

[I-D.ietf-netmod-sub-intf-vlan-model]
Wilton, R., Ball, D., tapsingh@cisco.com, t., and S. Sivaraj, "Sub-interface VLAN YANG Data Models", draft-ietf-netmod-sub-intf-vlan-model-03 (work in progress), October 2017.

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

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

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

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

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

Authors' Addresses

Robert Wilton (editor)
Cisco Systems

Email: rwilton@cisco.com

David Ball
Cisco Systems

Email: daviball@cisco.com

Tapraj Singh
Cisco Systems

Email: tapsingh@juniper.net

Selvakumar Sivaraj
Juniper Networks

Email: ssivaraj@juniper.net

Network Working Group
Internet-Draft
Updates: rfc6087bis (if approved)
Intended status: Standards Track
Expires: April 20, 2019

C. Hopps
Deutsche Telekom
L. Berger
LabN Consulting, L.L.C.
D. Bogdanovic

October 17, 2018

YANG Module Tags
draft-ietf-netmod-module-tags-03

Abstract

This document provides for the association of tags with YANG modules. The expectation is for such tags to be used to help classify and organize modules. A method for defining, reading and writing a modules tags is provided. Tags may be standardized and assigned during module definition; assigned by implementations; or dynamically defined and set by users. This document provides guidance to future model writers and, as such, this document updates [I-D.ietf-netmod-rfc6087bis].

Status of This Memo

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

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

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

This Internet-Draft will expire on April 20, 2019.

Copyright Notice

Table of Contents

1. Introduction

The use of tags for classification and organization is fairly ubiquitous not only within IETF protocols, but in the internet itself (e.g., #hashtags). Tags can be usefully standardized, but they can also serve as a non-standardized mechanism available for users to define themselves. Our solution provides for both cases allowing for the most flexibility. In particular, tags may be standardized as well as assigned during module definition; assigned by implementations; or dynamically defined and set by users.

This document defines a YANG module [RFC6020] which provides a list of module entries to allow for adding or removing of tags as well as viewing the set of tags associated with a module.

This document defines an extension statement to be used to indicate tags that SHOULD be added by the module implementation automatically (i.e., outside of configuration).

This document also defines an IANA registry for tag prefixes as well as a set of globally assigned tags.

Section 7 provides guidelines for authors of YANG data models. This section updates [I-D.ietf-netmod-rfc6087bis].

2. Conventions Used in This Document

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

3. Tag Values

All tags begin with a prefix indicating who owns their definition. An IANA registry is used to support standardizing tag prefixes. Currently 3 prefixes are defined with all others reserved. No further structure is imposed by this document on the value following the standard prefix, and the value can contain any yang type ’string’ characters except carriage-returns, newlines and tabs.

3.1. IETF Standard Tags

An IETF standard tag is a tag that has the prefix "ietf:". All IETF standard tags are registered with IANA in a registry defined later in this document.

3.2. Vendor Tags

A vendor tag is a tag that has the prefix "vendor:". These tags are defined by the vendor that implements the module, and are not standardized; however, it is RECOMMENDED that the vendor include extra identification in the tag to avoid collisions such as using the enterpise or organization name follwing the "vendor:" prefix (e.g., vendor:example.com:vendor-defined-classifier).

3.3. User Tags

A user tag is any tag that has the prefix "user:". These tags are defined by the user/administrator and will never be standardized.

3.4. Reserved Tags

Any tag not starting with the prefix "ietf:", "vendor:" or "user:" is reserved for future standardization.

4. Tag Management

Tags can become associated with a module in a number of ways. Tags may be defined and associated at module design time, at implementation time, or via user administrative control. As the main consumer of tags are users, users may also remove any tag, no matter how the tag became associated with a module.

4.1. Module Definition Association

A module definition can indicate a set of tags to be added by the module implementer. These design time tags are indicated using the module-tag extension statement. If the module definition will be IETF standards track, the tags MUST also be IETF standard tags (Section 3.1). Thus, new modules can drive the addition of new standard tags to the IANA registry, and the IANA registry can serve as a check against duplication.

4.2. Implementation Association

An implementation MAY include additional tags associated with a module. These tags may be standard or vendor specific tags.

4.3. Administrative Tagging

Tags of any kind can be assigned and removed with using normal configuration mechanisms.

5. Tags Module Structure

5.1. Tags Module Tree

The tree associated with the "ietf-module-tags" module follows. The meaning of the symbols can be found in [RFC8340].

```
module: ietf-module-tags
  +--rw module-tags
     +--rw module* [name]
        +--rw name          yang:yang-identifier
        +--rw tag*          tag
        +--rw masked-tag*   tag
```

5.2. Tags Module

```
<CODE BEGINS> file "ietf-module-tags@2018-10-17.yang"
module ietf-module-tags {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-module-tags";
  prefix tags;

  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF NetMod Working Group (NetMod)";
  contact
    "NetMod Working Group - <netmod@ietf.org>";

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

  description
    "This module describes a mechanism associating tags with YANG
     modules. Tags may be IANA assigned or privately defined.

     Copyright (c) 2018 IETF Trust and the persons identified as
     authors of the code. All rights reserved.

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

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
     NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and
     'OPTIONAL' in the module text are to be interpreted as described
     in RFC 2119 (https://tools.ietf.org/html/rfc2119).

     This version of this YANG module is part of RFC XXXX
     (https://tools.ietf.org/html/rfcXXXX); see the RFC itself for
     full legal notices.";

  // RFC Ed.: update the date below with the date of RFC publication
  // and RFC number and remove this note.

  revision 2018-10-17 {
    description
```

```
      "Initial revision.";
    reference "RFC XXXX: YANG Module Tags";
  }

  typedef tag {
    type string {
      length "1..max";
      pattern '[a-zA-Z_][a-zA-Z0-9\-_]*:[\S ]+';
    }
    description
      "A tag value is composed of a standard prefix followed by any type
       'string' value that does not include carriage return, newline or
       tab characters.";
  }

  extension module-tag {
    argument tag;
    description
      "The argument 'tag' is of type 'tag'. This extension statement is
       used by module authors to indicate the tags that SHOULD be added
       automatically by the system. As such the origin of the value
       for the pre-defined tags should be set to 'system'.";
  }

  container module-tags {
    description
      "Contains the list of modules and their associated tags";
    list module {
      key "name";
      description
        "A list of modules and their associated tags";
      leaf name {
        type yang:yang-identifier;
        mandatory true;
        description
          "The YANG module name.";
      }
      leaf-list tag {
        type tag;
        description
          "Tags associated with the module. See the IANA 'YANG Module
           Tag Prefix' registry for reserved prefixes and the IANA 'YANG
           Module IETF Tag' registry for IETF standard tags.

           The operational view of this list is constructed using the following
steps:

           1) System added tags are added.
           2) User configured tags are added.
```

```
            3) Any tag that is equal to a masked-tag is removed.";
        }
        leaf-list masked-tag {
          type tag;
          description
            "The list of tags that should not be associated with this
             module. This user can remove (mask) tags by adding
             them to this list. It is not an error to add tags to this
             list that are not associated with the module.";
        }
      }
    }
}
<CODE ENDS>
```

6. Other Classifications

It's worth noting that a different YANG module classification document exists [RFC8199]. That document is classifying modules in only a logical manner and does not define tagging or any other mechanisms. It divides YANG modules into 2 categories (service or element) and then into one of 3 origins: standard, vendor or user. It does provide a good way to discuss and identify modules in general. This document defines standard tags to support [RFC8199] style classification.

7. Guidelines to Model Writers

This section updates [I-D.ietf-netmod-rfc6087bis].

7.1. Define Standard Tags

A module can indicate using module-tag extension statements a set of tags that are to be automatically associated with it (i.e., not added through configuration).

```
module example-module {
  ...
  import module-tags { prefix tags; }

  tags:module-tag "ietf:some-new-tag";
  tags:module-tag "ietf:some-other-tag";
  ...
}
```

The module writer can use existing standard tags, or use new tags defined in the model definition, as appropriate. For standardized

modules new tags MUST be assigned in the IANA registry defined below, see Section 8.2 below.

8. IANA Considerations

8.1. YANG Module Tag Prefix Registry

This registry allocates tag prefixes. All YANG module tags SHOULD begin with one of the prefixes in this registry.

The allocation policy for this registry is Specification Required [RFC5226].

The initial values for this registry are as follows.

prefix	description
ietf:	IETF Standard Tag allocated in the IANA YANG Module IETF Tag Registry.
vendor:	Non-standardized tags allocated by the module implementer.
user:	Non-standardized tags allocated by and for the user.

Other SDOs (standard organizations) wishing to standardize their own set of tags could allocate a top level prefix from this registry.

8.2. YANG Module IETF Tag Registry

This registry allocates prefixes that have the standard prefix "ietf:". New values should be well considered and not achievable through a combination of already existing standard tags.

The allocation policy for this registry is IETF Review [RFC5226].

The initial values for this registry are as follows.

Tag	Description	Reference
ietf:rfc8199-element	A module for a network element.	[RFC8199]
ietf:rfc8199-service	A module for a network service.	[RFC8199]
ietf:rfc8199-standard	A module defined by a standards organization.	[RFC8199]
ietf:rfc8199-vendor	A module defined by a	[RFC8199]

	vendor.	
ietf:rfc8199-user	A module defined by the user.	[RFC8199]
ietf:hardware	A module relating to hardware (e.g., inventory).	[This document]
ietf:software	A module relating to software (e.g., installed OS).	[This document]
ietf:qos	A module for managing quality of service.	[This document]
ietf:protocol	A module representing a protocol.	[This document]
ietf:system-management	A module relating to system management (e.g., a system management protocol such as syslog, TACAC+, SNMP, netconf, ...).	[This document]
ietf:network-service	A module relating to network service (e.g., a network service protocol such as an NTP server, DNS server, DHCP server, etc).	[This document]
ietf:oam	A module representing Operations, Administration, and Maintenance (e.g., BFD).	[This document]
ietf:routing	A module related to routing.	[This document]
ietf:signaling	A module representing control plane signaling.	[This document]
ietf:lmp	A module representing a link management protocol.	[This document]

Table 1: IETF Module Tag Registry

9. References

9.1. Normative References

[I-D.ietf-netmod-rfc6087bis]
Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-20 (work in progress), March 2018.

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

[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>.

[RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <https://www.rfc-editor.org/info/rfc8199>.

9.2. Informative References

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

Authors' Addresses

Christan Hopps
Deutsche Telekom

Email: chopps@chopps.org

Lou Berger
LabN Consulting, L.L.C.

Email: lberger@labn.net

Dean Bogdanovic

Email: ivandean@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2019

A. Clemm
Y. Qu
Huawei
J. Tantsura
Nuage Networks
A. Bierman
YumaWorks

October 21, 2018

Comparison of NMDA datastores
draft-ietf-netmod-nmda-diff-00

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on April 24, 2019.

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

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

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

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

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

2. Key Words

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

3. Definitions and Acronyms

NMDA: Network Management Datastore Architecture

RPC: Remote Procedure Call

4. Data Model Overview

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

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

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

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

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

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

The operation provides the following output parameter:

- differences: This parameter contains the list of differences, encoded per RFC8072, i.e. specifying which patches would need to be applied to the source to produce the target. When the target datastore is <operational>, "origin" metadata is included as part of the patch. Including origin metadata can help explain the

cause of a difference, for example when a data node is part of <intended> but the origin of the same data node in <operational> is reported as "system".

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

```
module: ietf-nmda-compare

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

Structure of ietf-nmda-compare

5. YANG Data Model

```
<CODE BEGINS> file "ietf-nmda-compare@2018-10-21.yang"
module ietf-nmda-compare {

  yang-version 1.1;
```

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

  prefix cp;

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

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

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

     Author: Yingzhen Qu
             <mailto:yingzhen.qu@huawei.com>

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

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

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

  revision 2018-10-21 {
    description
      "Initial revision";
    reference
      "RFC XXXX: Comparison of NMDA datastores";
  }

  /* RPC */
  rpc compare {
    description
```

```
      "NMDA compare operation.";
    input {
      leaf source {
        type identityref {
          base ds:datastore;
        }
        mandatory true;
        description
          "The source datastore to be compared.";
      }
      leaf target {
        type identityref {
          base ds:datastore;
        }
        mandatory true;
        description
          "The target datastore to be compared.";
      }
      leaf all {
            type empty;
        description
          "When this leaf is provided, all data nodes are compared,
                   whether their schema node pertains to both datastores or
           not. When this leaf is omitted, a prefiltering step is
           automatically applied that excludes data nodes from the
           comparison that can occur in only one datastore but not
           the other.  Specifically, if one of the datastores
           (source or target) contains only configuration data and
           the other datastore is <operational>, data nodes for
           which config is false are excluded from the comparison.";
      }
      choice filter-spec {
        description
          "Identifies the portions of the datastores to be
               compared.";
        anydata subtree-filter {
          description
            "This parameter identifies the portions of the
             target datastore to retrieve.";
          reference "RFC 6241, Section 6.";
        }
        leaf xpath-filter {
          if-feature nc:xpath;
          type yang:xpath1.0;
          description
            "This parameter contains an XPath expression
             identifying the portions of the target
             datastore to retrieve.";
```

```
        }
      }
    }
    output {
      choice compare-response {
        leaf no-matches {
          type empty;
          description
            "This leaf indicates that the filter did not match anything
             and nothing was compared.";
        }
        container differences {
          uses ypatch:yang-patch;
          description
            "The list of differences, encoded per RFC8072.";
        }
        description
          "Comparision results.";
      }
    }
  }
}
<CODE ENDS>
```

6. Example

The following example compares the difference between <operational> and <intended> for object "explicit-router-id", as defined in data module [I-D.ietf-ospf-yang].

RPC request:

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

RPC reply, when a difference is detected:

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

RPC reply when no difference is detected:

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

The same request in RESTCONF (using JSON format):

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

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

The same response in RESTCONF (using JSON format):

```
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+json

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

7. Open Issues

Currently, origin metadata is included in RPC output per default in comparisons that involve <operational>. It is conceivable to introduce an input parameter that controls whether origin metadata should in fact be included.

Currently the comparison filter is defined using subtree and XPath as in NETCONF[RFC6241]. It is not clear whether there is a requirement to allow for the definition of filters that relate instead to target resources per RESTCONF [RFC7950].

8. Possible Future Extensions

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

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

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

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

9. IANA Considerations

9.1. Updates to the IETF XML Registry

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

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

Registrant Contact: The IESG.

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

9.2. Updates to the YANG Module Names Registry

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

name: ietf-nmda-compare

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

prefix: cp

reference: RFC XXXX

10. Security Considerations

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

11. Acknowledgments

We thank Rob Wilton, Martin Bjorklund, Mahesh Jethanandani, Lou Berger, Kent Watsen, Phil Shafer, Ladislav Lhotka for valuable feedback and suggestions on an earlier revision of this document.

12. References

12.1. Normative References

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

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

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

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

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

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

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

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

[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>.

12.2. Informative References

[I-D.ietf-ospf-yang] Yeung, D., Qu, Y., Zhang, Z., Chen, I., and A. Lindem, "YANG Data Model for OSPF Protocol", draft-ietf-ospf-yang-17 (work in progress), September 2018.

Authors' Addresses

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

Email: ludwig@clemm.org

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

Email: yingzhen.qu@huawei.com

Jeff Tantsura
Nuage Networks

Email: jefftant.ietf@gmail.com

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 3, 2019

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

July 2, 2018

Sub-interface VLAN YANG Data Models
draft-ietf-netmod-sub-intf-vlan-model-04

Abstract

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

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.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 3, 2019.

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, such as IPv6 [RFC2460], Ethernet Pseudo Wires [RFC4448] and VPLS [RFC4761] [RFC4762] to be configurable via YANG when interoperating with VLAN tagged traffic received from an IEEE 802.1Q compliant bridge.

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

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

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

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

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

1.1. Terminology

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

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

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

1.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration (read-write), and "ro" means state data (read-only).
- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list or leaf-list.
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

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.

2.2. Extensibility

The modules are structured in such a way that they can be sensibly extended. In particular:

The tag stack is represented as a list to allow a tag stack of more than two tags to be supported if necessary in future.

The tag stack list elements allow other models, or vendors, to include additional forms of tag matching and rewriting. The intention, however, is that it should not be necessary to have any vendor specific extensions to any of the YANG models defined in this document to implement standard Ethernet and VLAN services.

3. L3 Interface VLAN Model

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

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

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

4. Flexible Encapsulation Model

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

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

The model supports flexible rewrites of the layer 2 frame header for data frames as they are processed on the interface. It defines a set of standard tag manipulations that allow for the insertion, removal, or rewrite of one or two 802.1Q VLAN tags. The expectation is that

manipulations are generally implemented in a symmetrical fashion, i.e. if a manipulation is performed on ingress traffic on an interface then the reverse manipulation is always performed on egress traffic out of the same interface. However, the model also allows for asymmetrical rewrites, which may be required to implement some forwarding models (such as E-Tree).

The structure of the model is currently limited to matching or rewriting a maximum of two 802.1Q tags in the frame header but has been designed to be easily extensible to matching/rewriting three or more VLAN tags in future, if required.

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

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

```
module: ietf-flexible-encapsulation
  augment /if:interfaces/if:interface/if-cmn:encapsulation/
                                                 if-cmn: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     ieee:vlanid
|     |           +--rw second-tag!
|     |              +--rw tag-type    dot1q-tag-type
|     |              +--rw vlan-id     ieee: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     ieee:vlanid
|        |        +--rw second-tag!
|        |           +--rw tag-type    dot1q-tag-type
|        |           +--rw vlan-id     ieee: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     ieee:vlanid
|                 +--rw second-tag!
|                    +--rw tag-type    dot1q-tag-type
|                    +--rw vlan-id     ieee:vlanid
|
+--rw local-traffic-default-encaps!
   +--rw outer-tag!
   |  +--rw tag-type    dot1q-tag-type
   |  +--rw vlan-id     ieee:vlanid
   +--rw second-tag!
      +--rw tag-type    dot1q-tag-type
      +--rw vlan-id     ieee:vlanid
```

5. L3 Interface VLAN YANG Module

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

```
<CODE BEGINS> file "ietf-if-l3-vlan@2017-10-30.yang"
module ietf-if-l3-vlan {
```

```
yang-version 1.1;

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

prefix if-l3-vlan;

import ietf-interfaces {
  prefix if;
}

import iana-if-type {
  prefix ianaift;
}

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

import ietf-interfaces-common {
  prefix if-cmn;
}

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

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

   WG Chair: Lou Berger
             <mailto:lberger@labn.net>

   WG Chair: Kent Watsen
             <mailto:kwatsen@juniper.net>

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

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

revision 2017-10-30 {
  description "Latest draft revision";

  reference
    "Internet-Draft draft-ietf-netmod-sub-intf-vlan-model-03";
}
```

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

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

       /*
        * Matches a single VLAN Id, or pair of VLAN Ids to classify
        * traffic into an L3 service.
        */
       case dot1q-vlan {
         container dot1q-vlan {
           must
             'count(../../if-cmn:forwarding-mode) = 0 or ' +
             'derived-from-or-self(../../if-cmn:forwarding-mode,' +
                                   '"if-cmn:layer-3-forwarding")' {
               error-message
                 "If the interface forwarding-mode leaf is set then it
                  must be set to an identity that derives from
                  layer-3-forwarding";

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


           description
             "Match VLAN tagged frames with specific VLAN Ids";
           container outer-tag {
             presence "The outermost VLAN tag exists";
```

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

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

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

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

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

            presence "The second outermost VLAN tag exists";

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

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

6. Flexible Encapsulation YANG Module

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

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

```
<CODE BEGINS> file "ietf-flexible-encapsulation@2017-10-30.yang"
module ietf-flexible-encapsulation {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-flexible-encapsulation";

  prefix flex;

  import ietf-interfaces {
    prefix if;
  }

  import iana-if-type {
    prefix ianaift;
  }

  import ietf-interfaces-common {
    prefix if-cmn;
  }

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

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

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

     WG Chair: Lou Berger
               <mailto:lberger@labn.net>

     WG Chair: Kent Watsen
               <mailto:kwatsen@juniper.net>

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

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

  revision 2017-10-30 {
    description "Latest draft revision";

    reference
```

```
        "Internet-Draft draft-ietf-netmod-sub-intf-vlan-model-03";
    }

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

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

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

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

        case default {
          description "Default match";
          leaf default {
            type empty;
```

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

        case untagged {
          description "Match untagged Ethernet frames only";
          leaf untagged {
            type empty;
            description
              "Untagged match.  Matches all untagged traffic.";
          } // leaf untagged
        } // case untagged

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

          container dot1q-priority-tagged {
            description "802.1Q priority tag match";
            leaf tag-type {
              type dot1q-types:dot1q-tag-type;
              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 {
              presence "The outermost VLAN tag exists";

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

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

            container second-tag {
              must
```

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

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

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

              presence "The second outermost VLAN tag exists";

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

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

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

    /*
     * Grouping for tag-rewrite that can be expressed either
     * symmetrically, or in the ingress and/or egress directions
     * independently.
     */
    grouping dot1q-tag-rewrite {
      description "Flexible rewrite";
      leaf pop-tags {
```

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

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

          container outer-tag {
            presence
              "Indicates existence of the outermost VLAN tag to
               push/rewrite";

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

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

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

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

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

            presence
              "Indicates existence of a second outermost VLAN tag to
               push/rewrite.";

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

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

```
    }

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

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

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

```
          description "Flexible encapsulation and rewrite";
          container flexible {
              must
                'count(../../if-cmn:forwarding-mode) = 0 or ' +
                'derived-from-or-self(../../if-cmn:forwarding-mode,' +
                                    '"if-cmn:layer-2-forwarding")' {
                error-message
                  "If the interface forwarding-mode leaf is set then it
                   must be set to an identity that derives from
                   layer-2-forwarding";

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

            description "Flexible encapsulation and rewrite";

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

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

                /*
                 * Allow asymmetrical rewrites to be specified.
                 */
                case asymmetrical {
                  if-feature asymmetric-rewrites;
                  description "Asymmetrical rewrite";
```

```
            container ingress {
              uses flexible-rewrite;
              description "Ingress rewrite";
            }
            container egress {
              uses flexible-rewrite;
              description "Egress rewrite";
            }
          }
        }
      }

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

        container outer-tag {
          presence
            "Indicates existence of the outermost VLAN tag";

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

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

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

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

          description
            "For IEEE 802.1Q interoperability, when specifying two
```

```
                   tags, it is required that the outermost tag exists and
                   is an S-VLAN, and the second outermost tag is a
                   C-VLAN";
                }

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

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

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

7. Examples

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

7.1. Layer 3 sub-interfaces with IPv6

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

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces
  xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
  xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
  xmlns:dot1q-types="urn:ieee:std:802.1Q:yang:ieee802-dot1q-types"
  xmlns:if-cmn="urn:ietf:params:xml:ns:yang:ietf-interfaces-common">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
    </interface>
    <interface>
      <name>eth0.1</name>
```

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

7.2. Layer 2 sub-interfaces with L2VPN

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

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

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces
  xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
  xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
  xmlns:dot1q-types="urn:ieee:std:802.1Q:yang:ieee802-dot1q-types"
  xmlns:if-cmn="urn:ietf:params:xml:ns:yang:ietf-interfaces-common">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
    </interface>
    <interface>
      <name>eth0.3</name>
      <type>ianaift:l2vlan</type>
      <if-cmn:parent-interface>eth0</if-cmn:parent-interface>
      <if-cmn:encapsulation>
        <flexible
    xmlns="urn:ietf:params:xml:ns:yang:ietf-flexible-encapsulation">
          <match>
            <dot1q-vlan-tagged>
              <outer-tag>
                <tag-type>dot1q-types:s-vlan</tag-type>
                <vlan-id>10</vlan-id>
              </outer-tag>
              <second-tag>
                <tag-type>dot1q-types:c-vlan</tag-type>
                <vlan-id>21</vlan-id>
              </second-tag>
            </dot1q-vlan-tagged>
          </match>
          <rewrite>
            <symmetrical>
              <dot1q-tag-rewrite>
                <pop-tags>2</pop-tags>
```

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

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

8. Acknowledgements

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

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

9. ChangeLog

9.1. WG version -04

- Added examples

9.2. WG version -03

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

9.3. WG version -02

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

9.4. WG version -01

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

9.5. Version -04

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

9.6. Version -03

- Incorporates feedback received from presenting to the IEEE 802.1 WG.
- Updates the modules for double tag matches/rewrites to restrict the outer tag type to S-VLAN and inner tag type to C-VLAN.
- Updates the introduction to indicate primary use case is for IETF forwarding protocols.
- Updates the objectives to make IEEE 802.1Q bridge interoperability a key objective.

10. IANA Considerations

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

11. Security Considerations

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

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

11.1. if-l3-vlan.yang

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

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

11.2. flexible-encapsulation.yang

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

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

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

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

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

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

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

12. References

12.1. Normative References

[I-D.ietf-netmod-intf-ext-yang]
Wilton, R., Ball, D., tsingh@juniper.net, t., and S. Sivaraj, "Common Interface Extension YANG Data Models", draft-ietf-netmod-intf-ext-yang-05 (work in progress), July 2017.

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

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

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

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

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

12.2. Informative References

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

[I-D.ietf-bess-l2vpn-yang] Shah, H., Brissette, P., Chen, I., Hussain, I., Wen, B., and K. Tiruveedhula, "YANG Data Model for MPLS-based L2VPN", draft-ietf-bess-l2vpn-yang-08 (work in progress), February 2018.

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

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

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

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

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

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

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

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 is also developing a YANG model for the configuration of 802.1Q VLANs. This raises the valid question as to whether the models overlap and whether it is necessary or beneficial to have two different models for superficially similar constructs. This section aims to answer that question by summarizing and comparing the two models.

A.1. Sub-interface based configuration model overview

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

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

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

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

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

- The model is flexible in the scope of the VLAN Identifier space. I.e. by default VLAN Ids can be scoped locally to a single

Ethernet-like trunk interface, but the scope is determined by the forwarding paradigm that is used.

A.2. IEEE 802.1Q Bridge Configuration Model Overview

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

- Each VLAN bridge component has a set of Ethernet interfaces that are members of that bridge. Sub-interfaces are not used, nor required in the 802.1Q bridge model.
- Within a VLAN bridge component, the VLAN tag in the packet is used, along with the destination MAC address, to determine how to forward the packet. Other forwarding paradigms are not supported by the 802.1Q model.
- Classification of traffic to a VLAN bridge component is based only on the Ethernet interface that it arrived on.
- VLAN Identifiers are scoped to a VLAN bridge component. Often devices only support a single bridge component and hence VLANs are scoped globally within the device.
- Feature configuration is specified in the context of the bridge, or particular VLANs on a bridge.

A.3. Possible Overlap Between the Two Models

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

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

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

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

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

Authors' Addresses

Robert Wilton (editor)
Cisco Systems

Email: rwilton@cisco.com

David Ball
Cisco Systems

Email: daviball@cisco.com

Tapraj Singh
Cisco Systems

Email: tapsingh@cisco.com

Selvakumar Sivaraj
Juniper Networks

Email: ssivaraj@juniper.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2018

A. Bierman
YumaWorks
M. Bjorklund
Tail-f Systems
K. Watsen
Juniper Networks
March 5, 2018

YANG Data Extensions
draft-ietf-netmod-yang-data-ext-01

Abstract

This document describes YANG mechanisms for defining abstract data structures with YANG. It is intended to replace and extend the "yang-data" extension statement defined in RFC 8040.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 6, 2018.

Copyright Notice

Table of Contents

1. Introduction

There is a need for standard mechanisms to allow the definition of abstract data that is not intended to be implemented as configuration or operational state. The "yang-data" extension statement from RFC 8040 [RFC8040] is defined for this purpose, however it is limited in its functionality.

The intended use of the "yang-data" extension is to model all or part of a protocol message, such as the "errors" definition in ietf-restconf.yang [RFC8040], or the contents of a file. However, protocols are often layered such that the header or payload portions of the message can be extended by external documents. The YANG statements that model a protocol need to support this extensibility that is already found in that protocol.

This document defines a new YANG extension statement called "augment-yang-data", which allows abstract data structures to be augmented from external modules, similar to the existing YANG "augment" statement. Note that "augment" cannot be used to augment a yang data structure since a YANG compiler or other tool is not required to understand the "yang-data" extension.

The "yang-data" extension from [RFC8040] has been copied here and updated to be more flexible. There is no longer a requirement for the "yang-data" statement to result in exactly one container object. There is no longer an assumption that a yang data structure can only be used as a top-level abstraction, instead of nested within some other data structure.

1.1. Terminology

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

The following terms are used within this document:

- yang data structure: A data structure defined with the "yang-data" statement.

1.1.1. NMDA

The following terms are defined in the Network Management Datastore Architecture (NMDA) [I-D.ietf-netmod-revised-datastores]. and are not redefined here:

- configuration
- operational state

1.1.2. YANG

The following terms are defined in [RFC7950]:

- absolute-schema-nodeid
- container
- data definition statement
- data node
- leaf
- leaf-list
- list

2. Definitions

2.1. Restrictions on Conceptual YANG Data

This document places restrictions on the "yang-data" external statements that can be used with the "yang-data" and "augment-yang-data" extensions. The conceptual data definitions are considered to be in the same identifier namespace as defined in section 6.2.1 of [RFC7950]. In particular, bullet 7:

> All leafs, leaf-lists, lists, containers, choices, rpcs, actions, notifications, anydatas, and anyxmls defined (directly or through a "uses" statement) within a parent node or at the top level of the module or its submodules share the same identifier namespace.

This means that conceptual data defined with the "yang-data" or "augment-yang-data" statements cannot have the same local-name as sibling nodes from regular YANG data definition statements or other "yang-data" or "augment-yang-data" statements.

This does not mean a yang data structure has to be used as a top-level protocol message or other top-level data structure. A yang data structure does not have to result in a single container.

2.2. YANG Data Extensions Module

The "ietf-yang-data-ext" module defines the "augment-yang-data" extension to augment conceptual data already defined with the "yang-data" extension. The RESTCONF "yang-data" extension has been moved to this document and updated.

RFC Ed.: update the date below with the date of RFC publication and remove this note.

<CODE BEGINS> file "ietf-yang-data-ext@2018-03-05.yang"

```
module ietf-yang-data-ext {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-data-ext";
  prefix "yd";

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

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

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

       Author:   Martin Bjorklund
                 <mailto:mbj@tail-f.com>

       Author:   Kent Watsen
                 <mailto:kwatsen@juniper.net>";


    description
      "This module contains conceptual YANG specifications
       for defining abstract 'yang-data' data structures.

       Copyright (c) 2017 - 2018 IETF Trust and the persons identified
       as authors of the code.  All rights reserved.

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

    revision 2018-03-05 {
      description
        "Initial revision.";
      reference
         // RFC Ed.: replace XXXX with RFC number and remove this note
        "RFC XXXX: YANG Data Extensions.";
    }


    extension yang-data {
      argument name {
        yin-element true;
      }
      description
        "This extension is used to specify a YANG data template which
         represents conceptual data defined in YANG. It is
         intended to describe hierarchical data independent of
         protocol context or specific message encoding format.
         Data definition statements within a yang-data extension
         specify the generic syntax for the specific YANG data
         template, whose name is the argument of the yang-data
         extension statement.

         Note that this extension does not define a media-type.
```

A specification using this extension MUST specify the message encoding rules, including the content media type.

The mandatory 'name' parameter value identifies the YANG data template that is being defined. It contains the template name. This parameter is only used for readability purposes. There are no mechanisms to reuse yang-data by its template name value.

This extension is ignored unless it appears as a top-level statement. It MUST contain data definition statements that result in a set of data definition statements.

If the yang data template is intended to be used as a top-level structure, then the yang data template needs to result in a single container, so an instance of the YANG data template can thus be translated into an XML instance document, whose top-level element corresponds to the top-level container.

The module name and namespace value for the YANG module using the extension statement is assigned to each of the data definition statements resulting from the yang data template. The name of each data definition statement resulting from a yang data template is assigned to a top-level identifier name in the data node identifier namespace, according to RFC 7950, section 6.2.1.

The sub-statements of this extension MUST follow the 'data-def-stmt' rule in the YANG ABNF.

The XPath document root is the extension statement itself, such that the child nodes of the document root are represented by the data-def-stmt sub-statements within this extension. This conceptual document is the context for the following YANG statements:

- must-stmt
- when-stmt
- path-stmt
- min-elements-stmt
- max-elements-stmt
- mandatory-stmt
- unique-stmt
- ordered-by
- instance-identifier data type

The following data-def-stmt sub-statements are constrained when used within a yang-data-resource extension statement.

```
          - The list-stmt is not required to have a key-stmt defined.
          - The if-feature-stmt is ignored if present.
          - The config-stmt is ignored if present.
          - The available identity values for any 'identityref'
            leaf or leaf-list nodes is limited to the module
            containing this extension statement, and the modules
            imported into that module.
      ";
  }


  extension augment-yang-data {
    argument path {
      yin-element true;
    }
    description
      "This extension is used to specify an augmentation to
       conceptual data defined with the 'yang-data' statement.
       It is intended to describe hierarchical data independent
       of protocol context or specific message encoding format.

       This statement has almost the same structure as the
       'augment-stmt'. Data definition statements within this
       statement specify the semantics and generic syntax for the
       additional data to be added to the specific YANG data template,
       identified by the 'path' argument.

       The mandatory 'path' parameter value identifies the YANG
       conceptual data node that is being augmented, represented
       as an absolute-schema-nodeid string.

       This extension is ignored unless it appears as a top-level
       statement. The sub-statements of this extension MUST follow
       the 'data-def-stmt' rule in the YANG ABNF.

       The module name and namespace value for the YANG module using
       the extension statement is assigned to instance document data
       conforming to the data definition statements within
       this extension.

       The XPath document root is the augmented extension statement
       itself, such that the child nodes of the document root are
       represented by the data-def-stmt sub-statements within
       the augmented yang-data statement.

       The context node of the augment-yang-data statement is derived
       in the same way as the 'augment' statement, as defined in
       section 6.4.1 of [RFC7950]. This conceptual node is
```

considered the context node for the following YANG statements:

- must-stmt
- when-stmt
- path-stmt
- min-elements-stmt
- max-elements-stmt
- mandatory-stmt
- unique-stmt
- ordered-by
- instance-identifier data type

The following data-def-stmt sub-statements are constrained when used within a augment-yang-data extension statement.

- The list-stmt is not required to have a key-stmt defined.
- The if-feature-stmt is ignored if present.
- The config-stmt is ignored if present.
- The available identity values for any 'identityref' leaf or leaf-list nodes is limited to the module containing this extension statement, and the modules imported into that module.

Example:

```
   foo.yang {
      import yang-data-ext { prefix yd; }

      yd:yang-data foo-data {
        container foo-con { }
      }
   }

   bar.yang {
      import yang-data-ext { prefix yd; }
      import foo { prefix foo; }

      yd:augment-yang-data /foo:foo-con {
        leaf add-leaf1 { type int32; }
        leaf add-leaf2 { type string; }
      }
   }
";
  }

 }
```

<CODE ENDS>

3. IANA Considerations

3.1. YANG Module Registry

This document registers one URI as a namespace in the "IETF XML Registry" [RFC3688]:

```
URI: urn:ietf:params:xml:ns:yang:ietf-yang-data-ext
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.
```

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

```
name:         ietf-yang-data-ext
namespace:    urn:ietf:params:xml:ns:yang:ietf-yang-data-ext
prefix:       yd
// RFC Ed.: replace XXXX with RFC number and remove this note
reference:    RFC XXXX
```

4. Security Considerations

This document defines YANG extensions that are used to define conceptual YANG data. It does not introduce any new vulnerabilities beyond those specified in YANG 1.1 [RFC7950].

5. Normative References

[I-D.ietf-netmod-revised-datastores] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-10 (work in progress), January 2018.

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

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

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

Appendix A. Examples

A.1. yang-data Example

This example shows a simple address book that could be stored as an artifact.

```
yd:yang-data example-address-book {
  container address-book {
    list address {
      key "last first";
      leaf last {
        type string;
        description "Last name";
      }
      leaf first {
        type string;
        description "First name";
      }
      leaf street {
        type string;
        description "Street name";
      }
      leaf city {
        type string;
        description "City name";
      }
      leaf state {
        type string;
        description "State name";
      }
    }
  }
}
```

A.2. augment-yang-data Example

This example adds "county" and "zipcode" leafs to the address book:

```
      yd:augment-yang-data /address-book/address {
        leaf county {
          type string;
          description "County name";
        }
        leaf zipcode {
          type string;
          description "Postal zipcode";
        }
      }
```

Appendix B. Change Log

B.1. v00 to v01

- moved open issues to github
- added examples section
- filled in IANA considerations

Appendix C. Open Issues

The YANG Data Extensions issues are tracked on github.com:

https://github.com/netmod-wg/yang-data-ext/issues

Authors' Addresses

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Kent Watsen
Juniper Networks

Email: kwatsen@juniper.net

NETMOD Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: April 19, 2019

K. Watsen
Juniper Networks
Q. Wu
Huawei Technologies
A. Farrel
Juniper Networks
B. Claise
Cisco Systems, Inc.
October 16, 2018

Handling Long Lines in Artwork in Internet-Drafts and RFCs
draft-kwatsen-netmod-artwork-folding-08

Abstract

This document introduces a simple and yet time-proven strategy for handling long lines in artwork in drafts using a backslash ('\') character where line-folding has occurred. The strategy works on any text based artwork, but is primarily intended for sample text and formatted examples and code, rather than for graphical artwork. The approach produces consistent results regardless of the content and uses a per-artwork header. The strategy is both self-documenting and enables automated reconstitution of the original artwork.

Table of Contents

1. Introduction

[RFC7994]sets out the requirements for plain-text RFCs and states that each line of an RFC (and hence of an Internet-Draft) must be

limited to 72 characters followed by the character sequence that denotes an end-of-line (EOL).

Internet-Drafts and RFCs often include example text or code fragments. In order to render the formatting of such text it is usually presented as a figure using the "<artwork>" element in the source XML. Many times the example text or code exceeds the 72 character line-length limit and the "xml2rfc" utility does not attempt to wrap the content of artwork, simply issuing a warning whenever artwork lines exceed 69 characters. According to the RFC Editor, there is currently no convention in place for how to handle long lines, other than advising authors to clearly indicate what manipulation has occurred.

This document introduces a simple and yet time-proven strategy for handling long lines using a backslash ('\') character where line-folding has occurred. The strategy works on any text based artwork, but is primarily intended for sample text and formatted examples and code, rather than for graphical artwork. The approach produces consistent results regardless of the content and uses a per-artwork header. The strategy is both self-documenting and enables automated reconstitution of the original artwork.

Note that text files are represent as lines having their first character in column 1, and a line length of N where the last character is in the Nth column and is immediately followed by an end of line character sequence.

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.

3. Goals

3.1. Automated Folding of Long Lines in Artwork

Automated folding of long lines is needed in order to support draft compilations that entail a) validation of source input files (e.g., XML, JSON, ABNF, ASN.1) and/or b) dynamic generation of output, using a tool that doesn't observe line lengths, that is stitched into the final document to be submitted.

Generally, in order for tooling to be able to process input files, the files must be in their original/natural state, which may include

having some long lines. Thus, these source files need to be modified before inclusion in the document in order to satisfy the line length limits. This modification SHOULD be automated to reduce effort and errors resulting from manual effort.

Similarly, dynamically generated output (e.g., tree diagrams) must also be modified, if necessary, in order for the resulting document to satisfy the line length limits. When needed, this effort again SHOULD be automated to reduce effort and errors resulting from manual effort.

3.2. Automated Reconstitution of Original Artwork

Automated reconstitution of the original artwork is needed to support validation of artwork extracted from documents. YANG [RFC7950] modules are already extracted from Internet-Drafts and validated as part of the draft-submission process. Additionally, there has been some discussion regarding needing to do the same for example YANG fragments contained within Internet-Drafts ([yang-doctors-thread]). Thus, it SHOULD be possible to mechanically reconstitute artwork in order to satisfy the tooling input parsers.

4. Limitations

4.1. Not Recommended for Graphical Artwork

While the solution presented in this document will work on any kind of text-based artwork, it is most useful on artwork that represents sourcecode (XML, JSON, etc.) or, more generally, on artwork that has not been laid out in two dimensions (e.g., diagrams).

Fundamentally, the issue is whether the artwork remains readable once folded. Artwork that is unpredictable is especially susceptible to looking bad when folded; falling into this category are most UML diagrams.

It is NOT RECOMMENDED to use the solution presented in this document on graphical artwork.

4.2. Doesn’t Work as Well as Format-Specific Options

The solution presented in this document works generically for all artwork, as it only views artwork as plain text. However, various formats sometimes have built-in mechanisms that can be used to prevent long lines.

For instance, both the ‘pyang‘ and ‘yanglint‘ utilities have the command line option "--tree-line-length" that can be used to indicate

a desired maximum line length for when generating tree diagrams [RFC8340].

In another example, some source formats (e.g., YANG [RFC7950]) allow any quoted string to be broken up into substrings separated by a concatenation character (e.g., '+'), any of which can be on a different line.

In yet another example, some languages allow factoring chunks of code into call outs, such as functions. Using such call outs is especially helpful when in some deeply-nested code, as they typically reset the indentation back to the first column.

As such, it is RECOMMENDED that authors do as much as possible within the selected format to avoid long lines.

5. Folded Structure

Artwork that has been folded as specified by this document MUST contain the following structure.

5.1. Header

The header is two lines long.

The first line is the following 46-character string that MAY be surrounded by any number of printable characters. This first line cannot itself be folded.

NOTE: '\\' line wrapping per BCP XX (RFC XXXX)

[Note to RFC Editor: Please replace XX and XXXX with the numbers assigned to this document and delete this note. Please make this change in multiple places in this document.]

The second line is a blank line. This line provides visual separation for readability.

5.2. Body

The character encoding is the same as described in Section 2 of [RFC7994], except that, per [RFC7991], tab characters are prohibited.

Lines that have a backslash ('\') occurring as the last character in a line immediately followed by the end of line character sequence, when the subsequent line starts with a backslash ('\') as the first non-space (' ') character, are considered "folded".

Really long lines may be folded multiple times.

6. Algorithm

6.1. Automated Folding

Determine the desired maximum line length from input. If no value is explicitly specified, the value "69" SHOULD be used.

Ensure that the desired maximum line length is not less than the minimum header, which is 46 characters. If the desired maximum line length is less than this minimum, exit (this artwork can not be folded).

Scan the artwork to see if any line exceeds the desired maximum. If no line exceeds the desired maximum, exit (this artwork does not need to be folded).

Scan the artwork for horizontal tab characters. If any horizontal tab characters appear, either resolve them to space characters or exit, forcing the input provider to convert them to space characters themselves first.

Scan the artwork to ensure no existing lines already end with a backslash ('\') character when the subsequent line starts with a backslash ('\') character as the first non-space (' ') character, as this would lead to an ambiguous result. If such a line is found, exit (this artwork cannot be folded).

For each line in the artwork, from top-to-bottom, if the line exceeds the desired maximum, then fold the line at the desired maximum column by 1) inserting the character backslash ('\') character at the maximum column, 2) inserting the end of line character sequence, inserting any number of space (' ') characters, and 4) inserting a further backslash ('\') character.

The result of this previous operation is that the next line starts with an arbitrary number of space (' ') characters, followed by a backslash ('\') character, immediately followed by the character that was previously in the maximum column.

Continue in this manner until reaching the end of the artwork. Note that this algorithm naturally addresses the case where the remainder of a folded line is still longer than the desired maximum, and hence needs to be folded again, ad infinitum.

6.1.1. Manual Folding

Authors may choose to fold text examples and source code by hand to produce a document that is more pleasant for a human reader but which can still be automatically unfolded (as described in Section 6.2) to produce single lines that are longer than the maximum document line length.

For example, an author may choose to make the fold at convenient gaps between words such that the backslash is placed in a lower column number than the artwork's maximum column value.

Additionally, an author may choose to indent the start of a continuation line by inserting space characters before the line continuation marker backslash character.

Manual folding may also help handle the cases that cannot be automatically folded as described in Section 6.

6.2. Automated Unfolding

All unfolding is assumed to be automated although a reader will mentally perform the act of unfolding the text to understand the true nature of the artwork or source code.

Scan the beginning of the artwork for the header described in Section 5.1. If the header is not present, starting on the first line of the artwork, exit (this artwork does not need to be unfolded).

Remove the 2-line header from the artwork.

For each line in the artwork, from top-to-bottom, if the line has a backslash ('\') character immediately followed by the end of line character sequence, and if the next line has a backslash ('\') character as the first non-space (' ') character, then the lines can be unfolded. Remove the first backslash ('\') character, the end of line character sequence, any leading space (' ') characters, and the second backslash ('\') character, which will bring up the next line. Then continue to scan each line in the artwork starting with the current line (in case it was multiply folded).

Continue in this manner until reaching the end of the artwork.

7. Considerations for xml2rfc v3

[RFC7991] introduces the vocabulary for version 3 of the xml2rfc tool. This includes a new element, "<sourcecode>" used to present sourcecode examples and fragments and to distinguish them from general artwork and in particular figures and graphics.

The folding and unfolding described in this document is applicable to the "<artwork>" element in both v2 and v3 of xml2rfc, and is equally applicable to the "<sourcecode>" element in xml2rfc v3.

8. Examples

The following self-documenting examples illustrate a folded document.

The source artwork cannot be presented here, as it would again need to be folded. Alas, only the result can be provided.

The examples in Sections 8.1 through 8.4 were automatically folded on column 69, the default value. Section 8.5 shows an example of manual folding.

8.1. Simple Example Showing Boundary Conditions

This example illustrates a boundary condition test using numbers for counting purposes. The input contains 5 lines, each line one character longer than the previous.

Any printable character (including ' ' and '\') can be used as a substitute for any number, except for on the 4th row, the trailing '9' is not allowed to be a '\' character if the first non-space character of the next line is a '\' character, as that would lead to an ambiguous result.

```
========== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) ===========

12345678901234567890123456789012345678901234567890123456789012345678
123456789012345678901234567890123456789012345678901234567890123456789
1234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901
1234567890123456789012345678901234567890123456789012345678901234567890123\
\90
12345678901234567890123456789012345678901234567890123456789012345678901234\
\901
123456789012345678901234567890123456789012345678901234567890123456789012345\
\9012
```

8.2. Example Showing Multiple Wraps of a Single Line

This example illustrates one very long line (280 characters).

Any printable character (including ' ' and '\') can be used as a substitute for any number.

```
========== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) ==========

1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678\
\90123456789012345678901234567890123456789012345678901234567890123456789012345\
\67890123456789012345678901234567890123456789012345678901234567890123456789012\
\34567890123456789012345678901234567890123456789012345678901234567890123456789\
\01234567890
```

8.3. Example With Native Backslash

This example has a '\' character in the wrapping column. The native text includes the sequence "fish\fowl" with the '\' character occurring on the 69th column.

```
string1="The quick brown dog jumps over the lazy dog which is a fish\
\\fowl as appropriate"
```

8.4. Example With Native Whitespace

This example has whitespace spanning the wrapping column. The native input contains 15 space (' ') characters between "like" and "white".

```
========== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) ==========

Sometimes our strings include multiple spaces such as "We like      \
\        white space."
```

8.5. Example of Manual Wrapping

This example was manually wrapped to cause the folding to occur after each term, putting each term on its own line. Indentation is used to additionally improve readability. Also note that the mandatory header is surrounded by different printable characters than shown in the other examples.

```
   [NOTE: '\\' line wrapping per BCP XX (RFC XXXX)]

   <request>::= <RP> \
                \<END-POINTS> \
                \[<LSPA>] \
                \[<BANDWIDTH>] \
                \[<metric-list>] \
                \[<RRO>[<BANDWIDTH>]] \
                \[<IRO>] \
                \[<LOAD-BALANCING>]
```

The manual folding produces a more readable result than the following equivalent folding that contains no indentation.

```
   ========== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) ==========

   <request>::= <RP> <END-POINTS> [<LSPA>] [<BANDWIDTH>] [<metric-list>\
   \] [<RRO>[<BANDWIDTH>]] [<IRO>] [<LOAD-BALANCING>]
```

9. Security Considerations

This BCP has no Security Considerations.

10. IANA Considerations

This BCP has no IANA Considerations.

11. References

11.1. Normative References

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

[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>.

11.2. Informative References

[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>.

[RFC7991] Hoffman, P., "The "xml2rfc" Version 3 Vocabulary", RFC 7991, DOI 10.17487/RFC7991, December 2016, <https://www.rfc-editor.org/info/rfc7991>.

[RFC7994] Flanagan, H., "Requirements for Plain-Text RFCs", RFC 7994, DOI 10.17487/RFC7994, December 2016, <https://www.rfc-editor.org/info/rfc7994>.

[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>.

[yang-doctors-thread] "[yang-doctors] automating yang doctor reviews", <https://mailarchive.ietf.org/arch/msg/yang-doctors/DCfBqgfZPAD7afzeDFlQ1Xm2X3g>.

Appendix A. POSIX Shell Script

This non-normative appendix section includes a shell script that can both fold and unfold artwork.

```
========== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) ==========

#!/bin/bash

print_usage() {
  echo
  echo "Folds the text file, only if needed, at the specified"
  echo "column, according to BCP XX."
  echo
  echo "Usage: $0 [-c <col>] [-r] -i <infile> -o <outfile>"
  echo
  echo "  -c: column to fold on (default: 69)"
  echo "  -r: reverses the operation"
  echo "  -i: the input filename"
  echo "  -o: the output filename"
  echo "  -d: show debug messages"
  echo "  -h: show this message"
  echo
  echo "Exit status code: zero on success, non-zero otherwise."
  echo
}


# global vars, do not edit
debug=0
reversed=0
infile=""
outfile=""
maxcol=69  # default, may be overridden by param
hdr_txt="NOTE: '\\\\' line wrapping per BCP XX (RFC XXXX)"
equal_chars="=============================================="
space_chars="                                              "

fold_it() {
  # since upcomming tests are >= (not >)
  testcol=`expr "$maxcol" + 1`

  # check if file needs folding
  grep ".\{$testcol\}" $infile >> /dev/null 2>&1
  if [ $? -ne 0 ]; then
    if [[ $debug -eq 1 ]]; then
      echo "nothing to do"
    fi
```

```
      cp $infile $outfile
      return -1
    fi

    foldcol=`expr "$maxcol" - 1` # for the inserted '\' char

    # ensure input file doesn't contain a TAB
    grep "\t" $infile >> /dev/null 2>&1
    if [ $? -eq 0 ]; then
      echo
      echo "Error: infile contains a TAB character, which is not allow\
\ed."
      echo
      return 1
    fi

    # ensure input file doesn't contain the fold-sequence already
    pcregrep -M  "\\\\\n[\ ]*\\\\" $infile >> /dev/null 2>&1
    if [ $? -eq 0 ]; then
      echo
      echo "Error: infile has a line ending with a '\' character follo\
\wed"
      echo "       by '\' as the first non-space character on the next\
\ line."
      echo "       This file cannot be folded."
      echo
      return 1
    fi

    # center header text
    length=`expr ${#hdr_txt} + 2`
    left_sp=`expr \( "$maxcol" - "$length" \) / 2`
    right_sp=`expr "$maxcol" - "$length" - "$left_sp"`
    header=`printf "%.*s %s %.*s" "$left_sp" "$equal_chars" "$hdr_txt"\
\ "$right_sp" "$equal_chars"`

    # fold using recursive passes ('g' didn't work)
    if [ -z "$1" ]; then
      # init recursive env
      cp $infile /tmp/wip
    fi
    gsed "/.\{$testcol\}/s/\(.\{$foldcol\}\)/\1\\\\\n\\\\/" < /tmp/wip\
\ >> /tmp/wip2
    diff /tmp/wip /tmp/wip2 > /dev/null 2>&1
    if [ $? -eq 1 ]; then
      mv /tmp/wip2 /tmp/wip
      fold_it "recursing"
    else
```

```
      echo "$header" > $outfile
      echo "" >> $outfile
      cat /tmp/wip2 >> $outfile
      rm /tmp/wip*
    fi

    ## following two lines represent a non-functional variant to the r\
  \ecursive
    ## logic presented in the block above.  It used to work before the\
  \ '\'
    ## on the next line was added to the format (i.e., the trailing '\\
  \\\\'
    ## in the substitution below), but now there is an off-by-one erro\
  \r.
    ## Leaving here in case anyone can fix it.
    #echo "$header" > $outfile
    #echo "" >> $outfile
    #gsed "/.\{$testcol\}/s/\(.\{$foldcol\}\)/\1\\\\\n\\\\/g" < $infil\
  \e >> $outfile

    return 0
  }


  unfold_it() {
    # check if file needs unfolding
    line=`head -n 1 $infile | fgrep "$hdr_txt"`
    if [ $? -ne 0 ]; then
      if [[ $debug -eq 1 ]]; then
        echo "nothing to do"
      fi
      cp $infile $outfile
      return -1
    fi

    # output all but the first two lines (the header) to wip (work in \
  \progress) file
    awk "NR>2" $infile > /tmp/wip

    # unfold wip file
    gsed ":x; /.*\\\\\$/N; s/\\\\\n[ ]*\\\\//; tx; s/\t//g" /tmp/wip >\
  \ $outfile

    # clean up and return
    rm /tmp/wip
    return 0
  }
```

```
   process_input() {
     while [ "$1" != "" ]; do
       if [ "$1" == "-h" -o "$1" == "--help" ]; then
         print_usage
         exit 1
       fi
       if [ "$1" == "-d" ]; then
         debug=1
       fi
       if [ "$1" == "-c" ]; then
         maxcol="$2"
         shift
       fi
       if [ "$1" == "-r" ]; then
         reversed=1
       fi
       if [ "$1" == "-i" ]; then
         infile="$2"
         shift
       fi
       if [ "$1" == "-o" ]; then
         outfile="$2"
         shift
       fi
       shift
     done

     if [ -z "$infile" ]; then
       echo
       echo "Error: infile parameter missing (use -h for help)"
       echo
       exit 1
     fi

     if [ -z "$outfile" ]; then
       echo
       echo "Error: outfile parameter missing (use -h for help)"
       echo
       exit 1
     fi

     if [ ! -f "$infile" ]; then
       echo
       echo "Error: specified file \"$infile\" is does not exist."
       echo
       exit 1
     fi
```

```
    min_supported=`expr ${#hdr_txt} + 8`
    if [ $maxcol -lt $min_supported ]; then
      echo
      echo "Error: the folding column cannot be less than $min_support\
  \ed"
      echo
      exit 1
    fi

    max_supported=`expr ${#equal_chars} + 1 + ${#hdr_txt} + 1 + ${#equ\
  \al_chars}`
    if [ $maxcol -gt $max_supported ]; then
      echo
      echo "Error: the folding column cannot be more than $max_support\
  \ed"
      echo
      exit 1
    fi

  }


  main() {
    if [ "$#" == "0" ]; then
       print_usage
       exit 1
    fi

    process_input $@

    if [[ $reversed -eq 0 ]]; then
      fold_it
      code=$?
    else
      unfold_it
      code=$?
    fi
    exit $code
  }

  main "$@"
```

Acknowledgements

The authors thank the following folks for their various contributions (sorted by first name): Jonathan Hansford, Joel Jaeggli, Lou Berger, Martin Bjorklund, Italo Busi, and Rob Wilton.

The authors additionally thank the RFC Editor, for confirming that there is no set convention today for handling long lines in artwork.

Authors' Addresses

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

Qin Wu
Huawei Technologies

EMail: bill.wu@huawei.com

Adrian Farrel
Juniper Networks

EMail: afarrel@juniper.net

Benoit Claise
Cisco Systems, Inc.

EMail: bclaise@cisco.com

Netconf
Internet-Draft
Intended status: Standards Track
Expires: April 22, 2019

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

October 19, 2018

YANG Based Instance Data Files Format
draft-lengyel-netmod-yang-instance-data-05

Abstract

There is a need to document data defined in YANG models without the need to fetch it from a live YANG server. Data is often needed already in design time or needed by groups that do not have a live running YANG server available. This document specifies a standard file format for YANG Based Instance data, that is data that could be stored in a datastore and whose syntax and semantics is defined by YANG models. Most important use cases foreseen include documenting server capabilities, factory-default settings, or vendor provided default configurations.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 22, 2019.

Table of Contents

1. Terminology

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

Design time: A time during which a YANG model and the implementation behind it is created. Sometimes in other documents this period is divided into design and implementation time.

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

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

Target YANG Module: A YANG module for which the instance data set contains instance data, like ietf-yang-library in the examples.

2. Introduction

There is a need to provide instance data defined in YANG models without the need to fetch it from a live YANG server. Data is often needed already in design time before the YANG server is implemented or needed by groups that do not have a live running YANG server available. To facilitate this off-line delivery of data this document specifies a standard file format for YANG Based Instance data, that is data that could be stored in a datastore and whose syntax and semantics is defined by YANG models.

2.1. Use Cases

We present a number of use cases were Yang based instance data is needed.

2.1.1. Use Case 1: Early Documentation of Server Capabilites

A YANG server has a number of server-capabilities that are defined in YANG modules and can be retrieved from the server using protocols like NETCONF or RESTCONF. YANG server capabilities include

- data defined in ietf-yang-library: YANG modules, submodules, features, deviations, schema-mounts, datastores supported ([I-D.ietf-netconf-rfc7895bis])
- alarms supported ([I-D.ietf-ccamp-alarm-module])
- data nodes, subtrees that support or do not support on-change notifications ([I-D.ietf-netconf-yang-push])
- netconf-capabilities in ietf-netconf-monitoring

While it is good practice to allow a client to query these capabilites from the live YANG server, that is often not enough.

Often when a network node is released an associated NMS (network management system) is also released with it. The NMS depends on the capabilities of the YANG server. During NMS implementation information about server capabilities is needed. If the information is not available early in some off-line document, but only as instance data from the live network node, the NMS implementation will be delayed, because it has to wait for the network node to be ready. Also assuming that all NMS implementors will have a correctly configured network node available to retrieve data from, is a very expensive proposition. (An NMS may handle dozens of node types.)

Network operators often build their own home-grown NMS systems that needs to be integrated with a vendor's network node. The operator needs to know the network node's server capabilities in order to do this. Moreover the network operator's decision to buy a vendor's product may even be influenced by the network node's OAM feature set documented as the Yang server's capabilites.

Beside NMS implementors, system integrators and many others also need the same information early. Examples could be model driven testing, generating documentation, etc.

Most server-capabilities are relatively stable and change only during upgrade or due to licensing or addition or removal of HW. They are usually defined by a vendor in design time, before the product is released. It feasible and advantageous to define/document them early e.g. in a Yang Based Instance Data File.

It is anticipated that a separate IETF document will define in detail how and which set of server capabilites should be documented.

2.1.2. Use Case 2: Preloading Data

There are parts of the configuration that must be fully configurable by the operator, however for which often a simple default configuration will be sufficient.

One example is access control groups/roles and related rules. While a sophisticated operator may define dozens of different groups often a basic (read-only operator, read-write system administrator, security-administrator) triplet will be enough. Vendors will often provide such default configuration data to make device configuration easier for an operator.

Defining Access control data is a complex task. To help the device vendor pre-defines a set of default groups (/nacm:nacm/groups) and rules for these groups to access specific parts of common models (/nacm:nacm/rule-list/rule).

YANG Based Instance data files are used to document and/or preload the default configurationp.

2.1.3. Use Case 3: Dcoumenting Factory Default Settings

Nearly every YANG server has a factory default configuration. If the system is really badly misconfigured or if the current configuration is to be abandoned the system can be reset to this default.

In Netconf the <delete-config> operation can already be used to do this for the startup configuration. There are ongoing efforts to introduce a new, more generic reset operation for the same purpose [I-D.wu-netconf-restconf-factory-restore]

The operator currently has no way to know what the default configuration actually contains. YANG Based Instance data can be used to document the factory default configuration.

3. Instance Data File Format

Two standard formats to represent YANG Based Instance Data are specified based on the XML and JSON encoding. The XML format is based on [RFC7950] while the JSON format is based on [RFC7951]. Later as other YANG encodings (e.g. CBOR) are defined further Instance Data formats may be specified.

For both formats data is placed in a top level auxiliary container named "instance-data-set". The purpose of the container, which is not part of the real data itself, is to carry meta-data for the complete instance-data-set.

The XML format SHALL follow the format returned for a NETCONF GET operation. The <data> anydata (which is not part of the real data itself) SHALL contain all data that would be inside the <data> wrapper element of a reply to the <get> operation. XML attributes SHOULD NOT be present, however if a SW receiving a YANG Based Instance data file encounters XML attributes unknown to it, it MUST ignore them, allowing them to be used later for other purposes.

The JSON format SHALL follow the format of the reply returmed for a RESTCONF GET request directed at the datastore resource: {+restconf}/data. ETags and Timestamps SHOULD NOT be included, but if present SHOULD be ignored.

A YANG Based Instance data file MUST contain a single instance data set. Instance data MUST conform to the corresponding target YANG Modules and follow the XML/JSON encoding rules as defined in [RFC7950] and [RFC7951] and use UTF-8 character encoding. A single instance data set MAY contain data for any number of target YANG modules, if needed it MAY carry the complete configuraton and state data set for a YANG server. Default values SHOULD NOT but MAY be included. Config=true and config=false data MAY be mixed in the instance data file. Instance data files MAY contain partial data sets. This means mandatory, min-elements or require-instance=true constrains MAY be violated.

The name of the file SHOULD be of the form:

instance-data-set-name ['@' revision-date] ('.yid')

E.g. acme-router-modules@2018-01-25.yid

The revision date is optional. It SHOULD NOT be used if the file is stored in a version control system (e.g. git) because the change of file names will break the connection between the different revisions of the file.

Meta data, information about the data set itself SHALL be included in the instance data set. This data will be children of the top level instance-data-set container as defined in the ietf-instance-data YANG module. Meta data SHALL include:

- Name of the instance data set

Meta data SHOULD include:

- Revision date of the instance data set
- Description of the instance data set. The description SHOULD contain information whether and how the data can change during the lifetime of the YANG server.

```
<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-router-modules</name>
  <revision>2108-01-25</revision>
  <description>Defines the minimal set of modules that any acme-router
       will contain. These modules will always be present.</description>
  <contact>info@acme.com</contact>
  <data>
    <yang-library xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
      <module-set>
        <name>basic</name>
        <module>
          <name>ietf-system</>
          <revision>2014-08-06</revision>
          <!-- description "A later revision may be used."; -->
          <namespace>urn:ietf:params:xml:ns:yang:ietf-system</namespace>
          <feature>authentication</feature>
          <feature>radius-authentication</feature>
        </module>
      </module-set>
    </yang-library>
  </data>
</instance-data-set>
```

Figure 1: XML Instance Data File example

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "acme-router-modules",
    "revision": "2108-01-25",
    "contact": "info@acme.com",
    "description":
      "Defines the set of modules that an acme-router will contain.",
    "data": {
      "ietf-yang-library:yang-library": {
        "module-set": [
          "name": "basic",
          "module": [
            {
              "name": "ietf-system",
              "revision": "2014-08-06",
              "namespace": "urn:ietf:params:xml:ns:yang:ietf-system",
              "feature": ["authentication", "radius-authentication"]
            }
          }
        ]
      ]
    }
  }
}
```

Figure 2: JSON Instance Data File example

4. Data Life cycle

Data defined or documented in YANG Based Instance Data Sets may be used for preloading a YANG server with this data, but the server may populate the data without using the actual file in which case the Instance Data File is only used as documentation.

While such data will usually not change, data documented by Instance Data sets MAY be changed by the YANG server itself or by management operations. It is out of scope for this document to specify a method to prevent this. Whether such data changes and if so, when and how, SHOULD be described either in the instance data file description statement or in some other implementation specific manner.

YANG Based Instance data is a snap-shot of information at a specific point of time. If the data changes afterwards this is not represented in the instance data set anymore, the valid values can be retrieved in run-time via Netconf/Restconf

Notifications about the change of data documented by Instance Data Sets may be supplied by e.g. the Yang-Push mechanism, but it is out of scope for this document.

5. Delivery of Instance Data

Instance data files SHOULD be available without the need for a live YANG server e.g. via download from the vendor's website, or any other way together with other product documentation.

6. YANG Model

<CODE BEGINS> file "ietf-yang-instance-data.yang"

```
module ietf-yang-instance-data {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data";
  prefix yid ;

  import ietf-yang-data-ext { prefix yd; }

  import ietf-datastores { prefix ds; }

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

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

  description "The module defines the structure and content of YANG
       Instance Data Sets.";

  revision 2018-06-30 {
    description  "Initial revision.";
    reference "RFC XXXX: YANG Based Instance Data";
  }

  yd:yang-data instance-data-format {
    container instance-data-set {
      description "Auxiliary container to carry meta-data for
        the complete instance data set.";

      leaf name {
        type string;
        mandatory true;
```

```
          description "Name of a YANG Based Instance data set.";
        }

        leaf description { type string; }

        leaf contact {
          type string;
          description "Contains the same information the contact
              statement carries for a YANG module.";
        }

        leaf organization {
          type string;
          description "Contains the same information the
              organization statement carries for a YANG module.";
        }

        leaf datastore {
          type ds:datastore-ref;
          description  "The identity of the datastore for which
            the instance data is documented for config=true data nodes.
            The leaf MAY be absent in which case the running dtastore or
            if thats not writable, the candidate datastore is implied.

            For config=false data nodes always the operational
            data store is implied.";
        }

        list revision {
          key date;
          description "An instance-data-set SHOULD have at least
              one revision entry.  For every published
              editorial change, a new one SHOULD be added in front
              of the revisions sequence so that all revisions are
              in reverse chronological order.";

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

          leaf description { type  string; }
        }

        anydata data {
```

```
          mandatory true;
          description "Contains the real instance data.
              The data MUST conform to the relevant YANG Modules.";
        }
      }
    }
  }
```

<CODE ENDS>

7. Security Considerations

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

8. IANA Considerations

To be completed, all the usual requests for a new YANG module

9. References

9.1. Normative References

[I-D.ietf-netmod-yang-data-ext]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Data Extensions", draft-ietf-netmod-yang-data-ext-01 (work in progress), March 2018.

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

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

9.2. Informative References

[I-D.ietf-ccamp-alarm-module]
Vallin, S. and M. Bjorklund, "YANG Alarm Module", draft-ietf-ccamp-alarm-module-04 (work in progress), October 2018.

[I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-07 (work in progress), October 2018.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", draft-ietf-netconf-yang-push-19 (work in progress), September 2018.

[I-D.wu-netconf-restconf-factory-restore]
Wu, Q., Lengyel, B., and Y. Niu, "Factory default Setting", draft-wu-netconf-restconf-factory-restore-03 (work in progress), October 2018.

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

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

Appendix A. Open Issues

- If we define metadata per target module, a list of target YAM could be included in the metadata. This depends on what additional metadata we will include.

- How do we know for which version of the target Yang Module is a data set valid? Proposal: One possibility would be to just indicate for which module version(s) was the data set last updated. This would be a hint about compatibility, but nothing more. Maybe we should wait till the YANG versioning work is complete/stable. Identifying just one version is way to strict, so something enforcing that shall not be used.

- Should we document what YANG features does the instance data set implicitly require? Proposal: that is already a use case, documenting data from the YANG library.

- Augmenting metadata must be possible. As of now it looks like yang-data-ext will solve that. If not, define instance data as regular YANG instead of yd:yang-data.

Appendix B. Changes between revisions

v04 - v05

- Changed title and introduction to clarify that this draft is only about the file format and documenting server capabilities is just a use case.
- Added reference to draft-wu-netconf-restconf-factory-restore
- Added new open issues.

v03 - v04

- Updated changelog for v02-v03

v02 - v03

- Updated the document according to comments received at IETF102
- Added parameter to specify datastore
- Rearranged chapters
- Added new use case: Documenting Factory Default Settings
- Added "Target YANG Module" to terminology
- Clarified that instance data is a snapshot valid at the time of creation, so it does not contain any later changes.
- Removed topics from Open Issues according to comments received at IETF102

v01 - v02

- The recommendation to document server capabilities was changed to be just the primary use-case. (Merged chapter 4 into the use case chapter.)
- Stated that RFC7950/7951 encoding must be followed which also defines (dis)allowed whitespace rules.
- Added UTF-8 encoding as it is not specified in t950 for instance data
- added XML declaration

v00 - v01

- Redefined using yang-data-ext

- Moved meta data into ordinary leafs/leaf-lists

Authors' Addresses

Balazs Lengyel
Ericsson
Magyar Tudosok korutja 11
1117 Budapest
Hungary

Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Phone: +32 2 704 5622
Email: bclaise@cisco.com

NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

N. Sambo
P. Castoldi
Scuola Superiore Sant'Anna
G. Fioccola
Huawei Technologies
F. Cugini
CNIT
H. Song
T. Zhou
Huawei

October 22, 2018

YANG model for finite state machine
draft-sambo-netmod-yang-fsm-04

Abstract

Network operators and service providers are facing the challenge of deploying systems from different vendors while looking for a trade-off among transmission performance, network device reuse, and capital expenditure without the need of being tied to single vendor equipment. The deployment and operation of more dynamic and programmable network infrastructures can be driven by adopting model-driven and software-defined control and management paradigms. In this context, YANG enables to compile a set of consistent vendor-neutral data models for networks and components based on actual operational needs emerging from heterogeneous use cases. This document proposes YANG models to describe events, operations, and finite state machine of YANG-defined network elements. The proposed models can be applied in several use cases: i) in the context of optical networks to pre-instruct data plane devices (e.g., an optical transponder) on the actions to be performed (e.g., code adaptation) in case some events, such as physical layer degradations, occur; ii) in general data networks, network telemetry applications can define and embed custom data probes into data plane devices. A probe in many cases can be modeled as an FSM; iii) the monitoring of packet loss and delay through a network clustering approach.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Networks are evolving toward more programmability, flexibility, and multi-vendor interoperability. Multi-vendor interoperability can be applied in the context of nodes, i.e. a node composed of components provided by different vendors (named fully disaggregated white box) is assembled under the same control system. This way, operators can optimize costs and network performance without the need of being tied to single vendor equipment. NETCONF protocol RFC6241 [RFC6241] based on YANG data modeling language RFC6020 [RFC6020] is emerging as a candidate Software Defined Networking (SDN) enabled protocol. First, NETCONF supports both control and management functionalities, thus permits high programmability. Then, YANG enables data modeling in a vendor-neutral way. Some recent works have provided YANG models to describe attributes of links (e.g., identification), nodes (e.g., connectivity matrix), media channels, and transponders (e.g., supported forward error correction - FEC) of networks ([I-D.ietf-i2rs-yang-network-topo] [I-D.vergara-ccamp-flexigrid-yang] [I-D.zhang-ccamp-l1-topo-yang]), also including optical technologies. This document presents YANG models to describe events, operations, and finite state machine of YANG-defined network elements. Such models can be applied to several use cases. In the context of elastic optical networks (EONs), the model enables a centralized remote network controller (managed by a network operator) to instruct a transponder controller about the actions to perform when certain events (e.g., failures) occur. The actions to be taken and the events can be re-programmed on the device. In general data networks, programmable network telemetry is considered a killer SDN application which can help applications gain unprecedented visibility to network data plane. Instead of providing raw data, network devices can be configured to filter and process data directly on the data plane and only hand preprocessed data to the collector, in order to save data bandwidth and reduce reaction delay ([I-D.song-opsawg-dnp4iq]) . Such configurations can be programed as custom probes and dynamically deployed into data plane devices. A probe in many cases can be modeled as an FSM. Another use case is the monitoring of packet loss and delay through a network clustering approach: in this case, each FSM state is determined by a specific subdivision of the network in Clusters ([I-D.fioccola-ippm-multipoint-alt-mark]).

2. Conventions used in this document

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

3. Terminology

ABNO: Application-Based Network Operations

BER: Bit Error Rate

EON: Elastic Optical Network

FEC: Forward Error Correction

FSM: Finite State Machine

NETCONF: Network Configuration Protocol

OAM: Operation Administration and Maintenance

SDN: Software Defined Network

YANG: Yet Another Network Generator

DNP: Dynamic Network Probe

AMM: Alternate Marking Method

4. Example of application

4.1. Pre-programming resiliency schemes in EONs

EONs (optical networks based on flexible grid supporting circuits of different bandwidth) are expected to employ flexible transponders, i.e. transponders supporting multiple bit rates, multiple modulation formats, and multiple codes. Such transponders permits the (re-) configuration of the bit rate value based on traffic requirements, as well as the configuration of the modulation format and code based on the physical characteristics of a path (e.g., quadrature phase shift keying is more robust than 16 quadrature amplitude modulation). This way, transmission parameters can be (re-) configured based on physical layer changes. The YANG model presented in this draft enables to pre-program reconfiguration settings of data plane devices in case of failures or physical layer degradations. In particular, soft failures are assumed. Soft failures imply transmission performance degradation, in turns a bit error rate (BER) increase, e.g. due to the ageing of some network devices. Without loosing generality, the ABNO architecture is assumed for the control and management of EONs (RFC7491 [RFC7491]). Considering the state of the art, when pre-FEC BER passes above a predefined threshold, it is expected that an alarm is sent to the OAM Handler, which communicates with the ABNO controller that may trigger an SDN controller (that

could be the Provisioning Manager of ABNO RFC7491 [RFC7491]) for computing new transmission parameters. The involved ABNO modules are shown in the simplified ABNO architecture of Fig. 1. Then, transponders are reconfigured. When alarms related to several connections impacted by the soft failure are generated, this procedure may be particularly time consuming. The related workflow for transponder reconfiguration is shown in Fig. 2. The proposed model enables an SDN controller to instruct the transponder about reconfiguration of new transmission parameters values if a soft failure occurs. This can be done before the failure occurs (e.g., during the connection instantiation phase or during the connection service), so that data plane devices can promptly reconfigure themselves without querying the SDN controller to trigger an on-demand recovery. This is expected to speed up the recovery process from soft failures. The related flow chart is shown in Fig. 3. The whole mechanism is based on a finite state machine where each state is associated to a specific configuration of transmission parameters (e.g., modulation format). The transition from a state to another state is triggered by specific events at the physical layer such as the bit error rate above a threshold. The transition from a state to another state implies a set of actions, including the change of transmission parameters (e.g., modulation format), which are actually suitable for the current condition at the physical layer. Moreover, since transmission and receiver must be synchronized about the transmission settings (modulation format and so no) for a proper transmission, another action consists of this synchronization. Thus, when the transponder at the receiver side decides to change its transmission parameters based on the monitored BER, the remote transponder at the transmitter side has to do the same state transition. In particular, the transponder at the receiver side sends a message to the transmitter to synchronize about the transmission parameters to be adopted. This message can be sent over a control channel. This way both the transmitter and receiver operates with the same transmission parameters: e.g. the format, FEC, and so on. No central controller is involved at this stage, only a notification can be sent to the central controller to inform it about the successful reconfiguration.

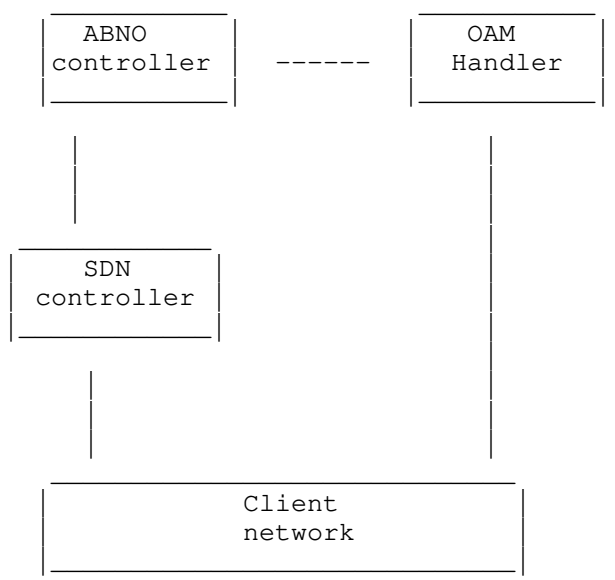


Figure 1: Assumed ABNO functional modules

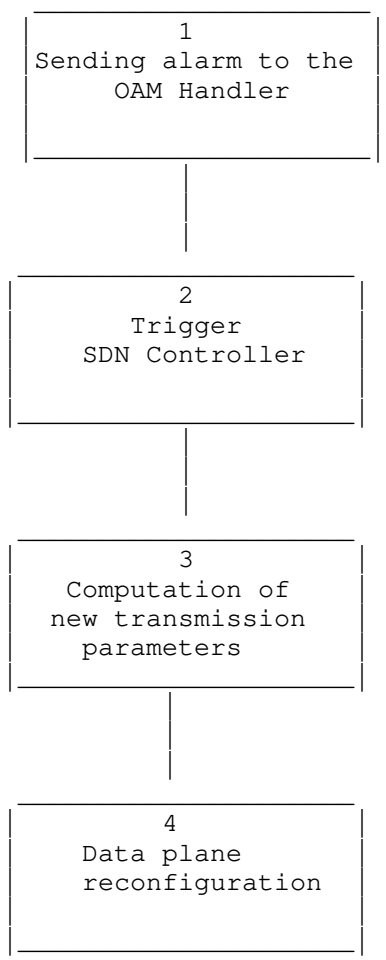


Figure 2: Flow chart of the expected state-of-the-art approach

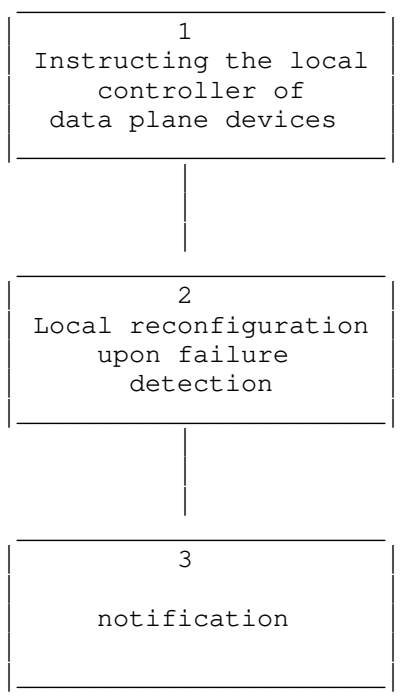


Figure 3: Flow chart of the approach exploiting YANG models in this draft

4.2. Deploying Dynamic Probes for Programmable Network Telemetry

In the past, network data analytics was considered a separate function from networks. They consume raw data extracted from networks through piecemeal protocols and interfaces. With the advent of user programmable data plane, we expect a paradigm shift that makes the data plane be an active component of the data telemetry and analytics solution. The programmable in-network data preprocessing is efficient and flexible to offload some light-weight data processing through dynamic data plane programming or configuration. A universal network data analytics platform built on top of this enables a tight and agile network control and OAM feedback loop. A proposed dynamic network telemetry system architecture is illustrated in Fig.4.

An application translates its data requirements into a set of Dynamic Network Probes (DNP) targeting a subset of data plane devices. After the probes are deployed, each probe conducts its corresponding in-network data preprocessing and feeds the preprocessed data to the

collector. The collector finishes the data post-processing and presents the results to the data-requesting application.

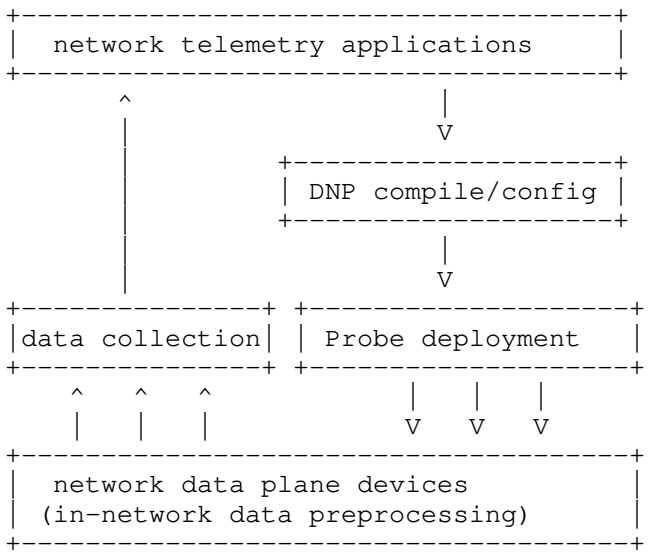


Figure 4: Deploy dynamic network probes using YANG FSM models

Many DNPs can be modeled as FSM which are configured to capture specific events. Here FSMs essentially preprocess the raw stream data and only report the necessary data to subscribing applications.

For example, a congestion control application needs to monitor the router buffer occupancy. Instead of polling the buffer depth periodically, it is only interested in the real-time events when the buffer depth crosses a low and a high threshold. We can install a probe to achieve this data plane function and the probe can be modeled as a three-state FSM. Each state represents a buffer region: below the low threshold, above the high threshold, and in between the two thresholds. A possible state transition is checked against the buffer depth for each incoming and outgoing packet. Whenever a state transition happens, an event is generated and reported to the application. This approach significantly reduces the amount of data sent to the application and also allows a timely event notification.

For another example, an application would like to monitor the delay experienced by a flow. The packet delay on its forwarding path can be acquired by using iOAM [I-D.brockners-inband-oam-requirements]. However, the application only needs to know that N consecutive flow packets experience a delay longer than T. Instead of forwarding the

raw delay data to the application, a probe can be deployed to detect the event. Similarly, the probe can be modeled as an FSM.

4.3. IP Performance Measurements on multipoint-to-multipoint large Networks

Networks offer rich sets of network performance measurement data, but traditional approaches run into limitations. One reason for this is the fact that in many cases, the bottleneck is the generation and export of the data and the amount of data that can be reasonably collected from the network runs into bandwidth and processing constraints in the network itself. In addition, management tasks related to determining and configuring which data to generate lead to significant deployment challenges.

In order to address these issues, an SDN controller application orchestrates network performance measurements tasks across the network to allow an optimized monitoring. In fact the IP Performance Measurement SDN Controller Application in Figure 5 can calibrate how deep can be obtained monitoring data from the network by configuring measurement points roughly or meticulously. This can be established by using the feedback mechanism reported in Figure 5.

For instance, the SDN Controller can configure initially an end to end monitoring between ingress points and egress points of the network. If the network does not experiment issues, this approximate monitoring is good enough and is very cheap in terms of network resources. But, in case of problems, the SDN Controller becomes aware of the issues from this approximate monitoring and, in order to localize the portion of the network that has issues, configures the measurement points more exhaustively. So a new detailed monitoring is performed. After the detection and resolution of the problem the initial approximate monitoring can be used again. This idea is general and can be applied to different performance measurements techniques both active and passive (and hybrid).

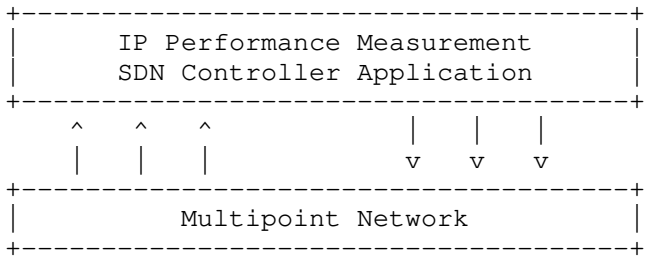


Figure 5: Feedback mechanism on multipoint-to-multipoint large Networks

One of the most efficient methodology to perform packet, loss delay and jitter measurements both in an IP and Overlay Networks is the Alternate Marking method, as presented in [I-D.ietf-ippm-alt-mark] and [I-D.fioccola-ippm-multipoint-alt-mark].

This technique can be applied to point-to-point flows but also to multipoint.to-multipoint flows (see [I-D.ietf-ippm-alt-mark] and [I-D.fioccola-ippm-multipoint-alt-mark]). The Alternate Marking method creates batches of packets by alternating the value of 1 or 2 bits of the packet header. These batches of packets are unambiguously recognized over the network and the comparison of packet counters permits the packet loss calculation. The same idea can be applied for delay measurement by selecting special packets with a marking bit dedicated for delay measurements. This method needs two counters each marking period for each flow under monitor. For this reason by considering n measurement points and n monitored flows, the order of magnitude of the packet counters for each time interval is n*n*2 (1 per color).

Multipoint Alternate Marking, described in [I-D.fioccola-ippm-multipoint-alt-mark], aims to reduce this value and makes the performance monitoring more flexible in case a detailed analysis is not needed.

It is possible to monitor a Multipoint Network without examining in depth by using the Network Clustering (subnetworks that are portions of the entire network that preserve the same property of the entire network). So in case there is packet loss or the delay is too high the filtering criteria could be specified more in order to perform a per flow detailed analysis, as described in [I-D.ietf-ippm-alt-mark].

An application of the multipoint performance monitoring can be done by using FSM (each state is a composition of clusters) and feedback mechanism where the SDN Controller is the brain of the network and can manage flow control to the switches and routers and, in the same way, can calibrate the performance measurements depending on the necessity.

5. YANG for finite state machine (FSM)

This model defines a list of states and transitions to describe a generic finite state machine (FSM). The related code and tree are shown in the Appendix.

<current-state>: it defines the current state of the FSM.
<states>: this element defines the FSM as follows.
<state>: this list defines all the FSM states.
<id>: this leaf attribute of <state> defines the

identifier of the state
<name>: this leaf attribute of <state> defines the name of the state
<description>: this leaf is a "string" describing the state
<transitions>: this attribute defines a list of transitions to other states in the FSM.
<name>: this attribute defines the name of a transition
<type>: this attribute defines the type of the transition from a pool of possible transition types predefined inside the YANG model. Together with the <name> attribute, it uniquely identifies the transition.
<description>: this optional attribute is a "string" describing the transition
<filters>: this leaf is a list of input parameters related to the transition. This attribute enables to further express a transition: as an example, if a transition can be triggered by a parameter (e.g., a monitored performance parameter) exceeding a threshold (as in Sec. 5), an element of the list defines this threshold. Thus, if the parameter is outside the threshold, the transition is taken, otherwise not.
<filter>: this leaf of <filters> defines a filter parameter.
<filter-id>: this leaf of <filters> define the identifier number associated with the <filter> attribute.
<actions>: this attribute defines a list of actions to take during the transition.
<action>: this attribute is the list of actions
<id>: this leaf of <action> defines the identifier number of an action.
<type>: this leaf of <action> defines the type of an action.
<simple>: this leaf defines (differently from <conditional> detailed below) an action that has to be directly executed.
<execute>: this attribute recalls an RPC encapsulating the effective task (action) to be executed by the

hardware. If more actions (e.g., "A" and "B"), defined in the <action> list, have to be executed, these actions can be executed sequentially according to the <next-action> attribute detailed below. Thus, by referring to the tree of the Appendix, when an action ("A") is executed, the <next-action> attribute will bring to another action ("B"). If more actions have to be executed in parallel (e.g., "A" & "B"), not sequentially, an element of the <action> list should be defined to express an action (e.g., "A&B") consisting of more actions to be executed in parallel.
<next-action>: this attribute defines the identification number of a next action that has to be taken. The <next-action> can assume a NULL value.
<conditional>: this leaf enables a check ("true" or "false") to be verified before executing the action. Based on the check, the proper attributes <execute> and <next-operation> are considered.
<statement>: this leaf of <conditional> defines the condition to be verified before executing the action.
<true>: this leaf of <conditional> defines a result of the check associated to <statement>. Proper <execute> and <next-operation> attributes are associated with this result of the

check.
<false>: this leaf of <conditional> defines a result of the check associated to <statement>. Proper <execute> and <next-operation> attributes are associated with this result of the check.
<next-state>: this attribute defines the next state of FSM when an action is executed.

6. Implementation of the pre-programming resiliency schemes in EONs

These presented model can be used to enable a centralized network controller, managed by a network operator, to instruct data plane hardware on its reconfiguration if some events, such as a failure or physical layer degradation, occur. As an example, an optical signal impacted by a soft failure (i.e., a physical layer degradation inducing a pre forward error correction bit error rate increase - pre-FEC) can be maintained by adapting the FEC of the signal itself. This action to be taken and, more in general operations to be executed depending on critical events, can be (re-) programmed on the transponder by (re-) sending a NETCONF <edit-config> message to the device controller including a FSM defined by the YANG model. Such a system has the main goal to speed up the reaction of the network to certain events/faults and to alleviate the workload of the centralized controller. The speed up derives from the fact that the centralized controller is able to pre-compute and pre-configure on the network devices the actions to take when an event occurs taking into account a global view and knowledge of the network. In this way, the device is already aware of the actions to be locally applied to reconfigure a connection, avoiding to inform the controller and to wait for the response indicating what to do. Consequently, part of the workload is also removed from the centralized controller. When the reaction is successfully completed in the data plane, the centralized controller can be notified about the faults and the taken action. A flexible transponder supporting two FEC types, 7% and 20%, is considered. A two-states FSM is also assumed. The states have <name> attribute set to "Steady" and "Fec-Baud-Adapt", respectively. In the "Steady" state, the signal is in a healthy condition, adopting a 7% FEC, with a pre-FEC BER below an assigned threshold of 9 x 10-4. A transition from this state can be triggered by the event with <name>=BER_CHANGE and <filter-type>=9 x 10-4, thus expressing a change of the pre-FEC BER above the threshold. In case the pre-FEC

BER exceeds 9 x 10-4 due to a soft failure, the state machine evolves to the "Fec-Baud-Adapt" state and an adaptation to a more robust FEC of 20% (executed by the attribute <execute>) is performed. The system can return to the "Steady" state if the pre-FEC BER goes below another pre-defined threshold and the FEC is reconfigured to 7%.

7. Appendix

This appendix reports the YANG models code and the related tree.

7.1. YANG model for FSM - Tree

```
module: ietf-fsm
   +--rw current-state?   leafref
   +--rw states
      +--rw state [id]
         +--rw id             state-id-type
         +--rw description?   string
         +--rw transitions
            +--rw transition [name type]
               +--rw name           string
               +--rw type           transition-type
               +--rw description?   string
               +--rw filters
               |  +--rw filter [filter-id]
               |     +--rw filter-id    uint32
               +--rw actions
                  +--rw action [id]
                     +--rw id             transition-id-type
                     +--rw type           enumeration
                     +--rw conditional
                     |  +--rw statement    string
                     |  +--rw true
                     |  |  +--rw execute
                     |  |  +--rw next-action?   transition-id-type
                     |  |  +--rw next-state?    leafref
                     |  +--rw false
                     |     +--rw execute
                     |     +--rw next-action?   transition-id-type
                     |     +--rw next-state?    leafref
                     +--rw simple
                        +--rw execute
                        +--rw next-action?   transition-id-type
                        +--rw next-state?    leafref
```

7.2. YANG model for FSM - Code

<CODE BEGINS> file "ietf-fsm@2016-03-15.yang"

```
 module ietf-fsm {
   namespace "http://sssup.it/fsm";
   prefix fsm;



   identity TRANSITION {
       description "Base for all types of event";
   }

   identity ON_CHANGE {
       base TRANSITION;
       description
         "The event when the database changes.";
   }

   // typedef statements

   typedef transition-type {
        description "it defines the type of transition (event)";
     type identityref {
       base TRANSITION;
```

```
     }

   }



   typedef transition-id-type {
        description "it defines the id of the transition (event)";


     type uint32;

   }




   // grouping statements

   grouping action-block {

        description "it defines the action to perform when a transition
        occurs";

     leaf id {

description "it refers to the id of the transition";
       type transition-id-type;

     }

     leaf type {

description "it defines if the action has to be simply executed or if
a conditional statement has to be checked before execution";

       type enumeration {

         enum "CONDITIONAL_OP" {
description "it defines the type CONDITIONAL OPERATION to check a
statement before execution";
}

         enum "SIMPLE_OP" {
description "it defines the type SIMPLE OPERATION: i.e., an operation
to be directly executed;
                }
```

```
        }
        mandatory true;
      }

      grouping execution-top {
         description "it defines the execution attribute";
        anyxml execute {
          description "Represent the action to perform";
        }
        leaf next-action {
          type transition-id-type;
          description "the id of the next action to execute";

        }
      }

      container conditional {
         description "it defines the container CONDITIONAL";
        when "../type = 'CONDITIONAL_OP'";
        leaf statement {
          type string;
          mandatory true;
          description
            "The statement to be evaluated before execution.
            E.g. if a=b";
```

```
       }

       container true {

description "it is referred to the result TRUE of a conditional
statement";

         uses execution-top;

       }

       container false {

description "it is referred to the result FALSE of a conditional
statement ";

         uses execution-top;

       }

     }



     container simple {
description "Simple execution of an action without checking any
condition";

       when "../type = 'SIMPLE_OP'";


       uses execution-top;

     }

   }



   grouping action-top {


description "it defines the grouping of action";

     list action {


description "it defines the list of actions";
```

```
        key "id";
        ordered-by user;
        uses action-block;
      }
    }

    grouping on-change {
      description
        "Event occuring when a modification of one or more
         objects occurs";

      container filters {
        description
          "This container contains a list of configurable filters
           that can be applied to subscriptions.  This facilitates
           the reuse of complex filters once defined.";
        list filter {
          key "filter-id";

          description
            "A list of configurable filters that can be applied to
             subscriptions.";
          leaf filter-id {
            type uint32;
            description
```

```
              "An identifier to differentiate between filters.";
          }
        }
      }
    }

   grouping transition-top {
description "it defines the grouping transition";
     leaf name {
description "it defines the transition name";
       type string;
       mandatory true;
     }

     leaf type {

description "it defines the transition type";
       type transition-type;
       mandatory true;
     }

     leaf description {

description "it describes the transition ";
       type string;
```

```
     }

     // list of all possible events
     uses on-change {
       when "type = 'ON_CHANGE'";
     }

     container actions {
description "it defines the container action";
       uses action-top;
     }
   }

   grouping transitions-top {
description "it defines the grouping transition";
     container transitions {
description "it defines the container transitions";
       list transition {
description "it defines the list of transitions";
         key "name type";
         uses transition-top;
       }
     }
```

```
   }

   // data definition statements

   uses transitions-top;

   // extension statements

   // feature statements

   // augment statements


   organization
     "Scuola Superiore Sant'Anna Network and Services Laboratory";

   contact
     " Editor: Matteo Dallaglio
               <mailto:m.dallaglio@sssup.it>
     ";

   description
     "This module contains a YANG definitions of a generic finite state
     machine.";
```

```
   revision 2016-03-15 {
     description "Initial Revision.";
     reference
       "RFC xxxx:";
   }

   // identity statements

   // typedef statements

   typedef state-id-type {
description "it defines the id type of the states";
     type uint32;
   }

   // grouping statements
   grouping state-top {
description "it defines the grouping state";
     leaf id {
description "it defines the id of a transition";
       type state-id-type;
     }
```

```
     leaf description {
description "it describes a transition";
       type string;
     }



     grouping next-state-top {
description "it defines the grouping for the next state";
       leaf next-state {
description "it defines the next state";
           type leafref {
description "it refers to its id";
             path "../../../../../../../../../states/state/id";
           }
           description "Id of the next state";
         }
     }

    uses transitions-top {
     augment "transitions/transition/actions/action/conditional/true" {
```

```
         uses next-state-top;

       }


     augment "transitions/transition/actions/action/conditional/false" {
         uses next-state-top;
       }
     augment "transitions/transition/actions/action/simple" {
         //uses next-state-top;
         leaf next-state {
description "it defines the next state";

           type leafref {
 description "it refers to its id";
             path "../../../../../../../../states/state/id";
           }
           description "Id of the next state";
         }
       }
    }


   }
```

```
   grouping states-top {

description "it defines the grouping states";

     leaf current-state {

description "it defines the current state";

       type leafref {
description "it refers to its id";
         path "../states/state/id";


       }

     }



     container states {
description "it defines the container states";

       list state {

   description "it defines the list of states";

         key "id";

         uses state-top;

       }

     }

   }




   // data definition statements
```

```
   uses states-top;

   // extension statements

   // feature statements

   // augment statements.

   // rpc statements

 }//module fsm
```

<CODE ENDS>

7.3. Example of values for the YANG model

FIELD NAME	YANG DATA TYPE	VALUE
Current State	leafref	"an existing state id in the FSM"
State		
id	uint32	1
name	string	Steady
description	string	"whatever string"
transition		
name	string	"whatever string"
type	enum	BER_CHANGE
description	string	"whatever string"
filter		
filter-id	uint32	2
filter-type	anyxml or xpath	BER>0.0009
action		
id	uint32	3
type	enum	SIMPLE
statement	string	"whatever string"
execute	anyxml	"this recalls an RPC where the FEC value is expressed"
next-operation	uint32	NULL
next-state	leafref	"an existing state id in the FSM"

8. Acknowledgements

This work has been partially supported by the European Commission through the H2020 ORCHESTRA (Optical peRformanCe monitoring enabling dynamic networks using a Holistic cross-layEr, Self-configurable Truly flexible approach, grant agreement no: H2020-645360) project. The views expressed here are those of the authors only. The European Commission is not liable for any use that may be made of the information in this document.

9. Other Contributors

Matteo Dallaglio (Scuola Superiore Sant'Anna), Andrea Di Giglio (Telecom Italia), Giacomo Bernini (Nextworks).

10. Security Considerations

TBD

11. IANA Considerations

TBD

12. References

12.1. Normative References

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

[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>.

[RFC7491] King, D. and A. Farrel, "A PCE-Based Architecture for Application-Based Network Operations", RFC 7491, DOI 10.17487/RFC7491, March 2015, <https://www.rfc-editor.org/info/rfc7491>.

12.2. Informative References

[I-D.brockners-inband-oam-requirements] Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi, T., <>, P., and r. remy@barefootnetworks.com, "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements-03 (work in progress), March 2017.

[I-D.fioccola-ippm-multipoint-alt-mark] Fioccola, G., Cociglio, M., Sapio, A., and R. Sisto, "Multipoint Alternate Marking method for passive and hybrid performance monitoring", draft-fioccola-ippm-multipoint-alt-mark-04 (work in progress), June 2018.

[I-D.ietf-i2rs-yang-network-topo]
Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A Data Model for Network Topologies", draft-ietf-i2rs-yang-network-topo-20 (work in progress), December 2017.

[I-D.ietf-ippm-alt-mark]
Fioccola, G., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate Marking method for passive and hybrid performance monitoring", draft-ietf-ippm-alt-mark-14 (work in progress), December 2017.

[I-D.song-opsawg-dnp4iq]
Song, H. and J. Gong, "Requirements for Interactive Query with Dynamic Network Probes", draft-song-opsawg-dnp4iq-01 (work in progress), June 2017.

[I-D.vergara-ccamp-flexigrid-yang]
Madrid, U., Perdices, D., Lopezalvarez, V., Dios, O., King, D., Lee, Y., and G. Galimberti, "YANG data model for Flexi-Grid Optical Networks", draft-vergara-ccamp-flexigrid-yang-06 (work in progress), January 2018.

[I-D.zhang-ccamp-l1-topo-yang]
zhenghaomian@huawei.com, z., Fan, Z., Sharma, A., and X. Liu, "A YANG Data Model for Optical Transport Network Topology", draft-zhang-ccamp-l1-topo-yang-07 (work in progress), April 2017.

Authors' Addresses

Nicola Sambo
Scuola Superiore Sant'Anna
Via Moruzzi 1
Pisa 56124
Italy

Email: nicola.sambo@sssup.it

Piero Castoldi
Scuola Superiore Sant'Anna
Via Moruzzi 1
Pisa 56124
Italy

Email: piero.castoldi@sssup.it

Giuseppe Fioccola
Huawei Technologies
Riesstrasse, 25
Munich 80992
Germany

Email: giuseppe.fioccola@huawei.com

Filippo Cugini
CNIT
Via Moruzzi 1
Pisa 56124
Italy

Email: filippo.cugini@cnit.it

Haoyu Song
Huawei
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: haoyu.song@huawei.com

Tianran Zhou
Huawei
156 Beiqing Road
Beijing 100095
China

Email: zhoutianran@huawei.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 25, 2019

R. Wilton
Cisco Systems, Inc.

October 22, 2018

YANG Versioning Potential Solutions
draft-verdt-netmod-yang-solutions-00

Abstract

This 'work in progress' document describes and evaluates potential solutions to the requirements stated in section 5 of the YANG versioning requirements draft. The aim of this draft is to only provide a progress update to the Netmod WG concerning the YANG versioning design team discussions on potential solutions, and to hopefully provide minimally sufficient information to allow the wider Netmod community to provide input into the direction of the YANG versioning design team.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 25, 2019.

Copyright Notice

Table of Contents

1. Terminology and Conventions

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

This document also makes use of the terminology introduced in the YANG versioning requirements draft (REF REQUIRED). In addition, this document introduces the following terminology:

- bc: Used as an abbreviation for a backwards-compatible change.

- nbc: Used as an abbreviation for a non-backwards-compatible change.

- editorial change: A backwards-compatible change that does not change the YANG module semantics in any way.

2. Introduction

This draft represents transient work in progress, and should be read as such. In particular, the descriptions of the solutions are not intended to be complete, nor necessarily consider all scenarios, but instead are intended to explore the broad approach and key aspects of the particular solution. The solution descriptions do not address all requirements at this time, instead they focus on the requirements that have the most significance on the final direction of the solution. Nor does this draft recommend any particular solution or solutions at this time. It is anticipated that once a final solution approach has been decided upon, that a separate draft shall be produced that will supersede this temporary draft.

The remainder of this document is split into the following sections:

Chapter Section 4 provides a condensed summary of the requirements, taken from [I-D.verdt-netmod-yang-versioning-reqs]. This section also lists where in the document these requirements are considered, if at all.

A significant part of this document is aimed at discussing the potential 'core' solutions, which are focussed on solving requirements: R1.1, R1.2, R1.4, R2.1, R2.2 and R4.4, and described in chapter Section 5.

Possible solutions for some of the secondary requirements, such as datanode lifecycle management, are considered in chapter Section 6. In particular, possible solutions for requirements R1.3, R4.1, R4.2 and R4.3 are considered.

Finally, chapter Section 7 lists some of the open issues that the YANG versioning design team are considering and working through. For some questions, a tentative design team direction of the answer is also given.

3. Background

Some members of the design team are authors of a potential solution draft to the YANG versioning requirements. The purpose of this document is to ensure that all reasonable solutions to the YANG

versioning problem have been properly considered before converging on a single chosen solution.

4. Summary of requirements

The requirement themselves are documented in section 5 of XXX. A shortened, non normative, summary of each of the requirements (using the same requirement numbers) is provided below to aid evaluation of the potential solutions.

Req 1.1 - MUST support nbc updates without breaking imports.

Req 1.2 - MUST support nbc updates without breaking existing client code.

Req 1.3 - MUST support import stmt restricted to only some revisions.

Req 1.4 - MUST support modules to be versioned by software release.

Req 2.1 - MUST be able to determine if two arbitrary versions of any MODULE are unchanged, bc, or nbc.

Req 2.2 - SHOULD be able to determine if two arbitrary versions of any DATA NODE are unchanged, bc, or nbc.

Req 3.1 - MUST allow servers to support existing clients.

Req 3.2 - MUST allow for simultaneously support of clients using different (perhaps restricted) revisions.

Req 4.1 - MUST provide way to indicate if deprecated nodes are implemented.

Req 4.2 - MUST be able to document reason for lifecycle changes, and possible alternative data nodes.

Req 4.3 - MUST be able to forewarn of future lifecycle changes.

Req 4.4 - SHOULD allow fixes to older revision of a module.

Req 5.1 - MUST provide guidance on how to use the new scheme.

Req 5.2 - MUST provide, and document, an upgrade path from existing YANG/protocols.

Req 5.3 - MUST consider versioning impact on instance data.

The following list indicates where solutions for particular requirements are considered in this draft.

Req 1.1 - Section 5, core solutions

Req 1.2 - Section 5, core solutions

Req 1.3 - Section 6, extra solutions

Req 1.4 - Section 5, core solutions

Req 2.1 - Section 5, core solutions

Req 2.2 - Section 5, core solutions

Req 3.1 - Deferred until main solution direction is chosen.

Req 3.2 - Deferred until main solution direction is chosen.

Req 4.1 - Section 6, extra solutions

Req 4.2 - Section 6, extra solutions

Req 4.3 - Section 6, extra solutions

Req 4.4 - Section 5, core solutions

Req 5.1 - Deferred until main solution direction is chosen.

Req 5.2 - Deferred untli main solution direction is chosen.

Req 5.3 - Deferred until main solution direction is chosen.

5. Potential solutions to core YANG versioning requirements

This section considers solutions that are aimed at solving the main YANG versioning requirements. In particular, the solutions described here are aimed at solving the following requirements: R1.1, R1.2, R1.4, R2.1, R2.2 and R4.4.

The solutions being considered are:

1. Module level 'major.minor.patch' semantic versioning

2. Module level 'major.minor.patch(x)' modified semantic versioning

3. Module level 'release.major.minor.patch' versioning

4. A tool based approach comparing YANG schema modules/trees

5. Follow existing RFC 7950 rules

5.1. Module level 'major.minor.patch' semantic versioning

This solution introduces a module level version number that adopts a subset of the semantic versioning rules published at semver.org.

The key part of this solution is a version number that comprises three fields, 'major.minor.patch':

1. major - updated only when a non-backwards-compatible change is made

2. minor - updated only when a backwards-compatible change is made

3. patch - updated only for 'editorial' changes that do not change the API semantics in any way

When a field in the version number is incremented, all following fields are reset back to 0. Major version number 0 indicates that the module is not yet stable and allows non-backwards-compatible changes without requiring the major version number to be incremented (e.g., this could be used in IETF drafts before they become RFCs).

If this solution is adopted, it is assumed that vendors would need to manage versioning of vendor YANG models independently of software release trains, and even then they would be limited in the scope of what changes are possible in an already shipped release, which is anticipated to not meet the business requirements of some vendors.

Solution advantages:

1. Follows widely known semantic versioning rules.

2. Version number alone indicates whether 2 module revisions are backwards-compatible.

3. Sufficient for most (but not necessarily all) YANG models developed by SDOs.

4. Matches the scheme being used by OpenConfig YANG models.

Solution disadvantages:

1. Does not fully support long lived vendor software release trains. In particular:

Does not necessarily allow for backwards-compatible changes (enhancements or fixes) in older releases.

Does not allow for non-backwards-compatible changes (enhancements or fixes) in older releases.

2. The 'patch' field is not as useful for YANG modules (which act like an API), since 'editorial' changes are likely to be less common that backwards-compatible enhancements and fixes.

5.2. Module level 'major.minor.patch(x)' modified semantic versioning

This solution modifies the semantic versioning solution described previously, with the principal aim of allow fixes to released code.

The change to the semantic versioning solution is a modification to how the 'patch' field is used. In addition to 'editorial' changes that do not change the YANG module semantics, the patch field can also be used in a limited way to indicate major and minor version changes as well. If the patch field is incremented for a minor version change that it is appended with the suffix '(m)', if the patch field is incremented for a major version change then it is appended with the suffix '(M)', replacing '(m)', if present. Once a given 'major.minor' version has a patch field value with '(m)' or '(M)' then all subsequent patch revisions on the same 'major.minor' version retain the letter '(m)' or '(M)' regardless of whether the subsequent changes are backwards-compatible, non-backwards-compatible, or editorial changes.

The updated semantic versioning rules for updating the 'major.minor.patch' version number is as follows:

1. if a non-backwards-compatible change is made then either the major version number MUST be updated (resetting the minor and patch version numbers to 0) or only the patch version number MUST be updated and appended with '(M)', replacing '(m)' if present.

2. if a backwards-compatible change is made then either the minor version number MUST be updated (resetting the patch version numbers to 0) or only the patch version number MUST be updated and appended with '(m)' unless the previous patch version number already had '(M)' appended, in which case the '(M)' suffix is retained for the new patch version.

3. if an editorial change is made then the patch version number MUST be updated. If the previous patch version number already had either an '(m') or '(M)' suffix then it is retained for the new patch version.

When a field in the version number is incremented, all following fields are reset back to 0. Major version number 0 indicates that the module is not yet stable and allows non-backwards-compatible changes without requiring the major version number to be incremented (e.g., this could be used in IETF drafts before they become RFCs).

If this solution is adopted, it is assumed that vendors would need to manage versioning of vendor YANG models independently of software release trains, but that they are able to release fixes to bugs in YANG module versions that are present in long lived software releases.

Where possible, the version number should be updated using the standard semantic versioning rules, relying on the '(m)' and '(M)' suffixes only used where strictly necessary.

Solution advantages:

1. Allows fixes to released YANG modules, whilst still preserving semver like semantics.

2. Aims to be sufficient for SDO and vendor YANG modules.

3. Modules can choose to just use semver rules if they wish. E.g. the scheme is compatible with the scheme being used by OpenConfig YANG models.

Solution disadvantages:

1. Slightly more complex than standard semver.org rules. The (m|M) suffix may be confusing, and their significance misinterpreted.

2. Within a 'major.minor' version branch it is not possible to determine whether a specific change is backwards-compatible or not.

3. If on a version with the (m) suffix, e.g. 'A.B.C(m)', it is not possible to determine whether an update to 'A.D.E', where D > B is a backwards-compatible change.

Variants:

Rather than using '(m)' or '(M)', it could instead use separate counters for bc and nbc changes, facilitating meaningful semantic versioning comparison between different patch versions on 'major.minor' branch.

Rather than overloading the patch version number, separate semantic version numbers could be used on branches. E.g. if a bc fix was required to version '1.2.3' this could be presented as '1.2.3/1.1.0', if there was a further nbc fix then the next branch version would be '1.2.3/2.0.0'.

5.3. Module level 'release.major.minor.patch' partial semantic versioning

This solution extends the semver 'major.minor.patch' version number scheme, by prefixing it with an explicit software release positive integer field.

The key part of this solution is a version number comprising four fields (release.major.minor.patch):

1. release - may be updated at any time (e.g. for a new major software release)

2. major - updated only when a non-backwards-compatible change is made

3. minor - updated only when a backwards-compatible change is made

4. patch - updated only only for changes that do not change the API semantics in any way

When a field in the version number is incremented, all following fields are reset back to 0, except for major that resets to 1. Release version number 0 indicates that the version is not yet stable and non-backwards-compatible changes are allowed without incrementing the major version number.

The assumption for this scheme is that the release number is always incremented for every major release, i.e. at any point where nbc changes may potentially be required in an older release.

Solution advantages:

1. Supports long lived vendor software release trains.

2. Completely allows bc and nbc changes (enhancements or fixes) in older independent releases.

3. Probably sufficient for YANG models developed by both vendors and SDOs.

Solution disadvantages:

1. Release version field must be incremented regardless of changes.

2. Version number is no longer an indicator of changes between 2 module revisions. I.e. the main benefit of semantic versioning is lost.

3. Differs from the scheme used by OpenConfig YANG model.

Similar variants:

The 'release' field could be regarded as optional, and if omitted, the version interpreted in exactly the same way as the module level 'major.minor.patch' semantic versioning solution.

5.4. A tool based approach comparing YANG schema modules/trees

This solution relies on using tooling to compare either two YANG modules, or two YANG schema trees to identify any changes between the two modules that do not conform to RFC 7950 section 11 backwards-compatibility rules.

Not all differences between two YANG statements in different module versions can easily be identified as backwards-compatible or not (for example changes in description, pattern statements, must or when statements may be hard to check). If a tool is unable to check then it would have to flag the change as potentially being non-backwards-compatible, potentially reporting many false positives.

To mitigate this, it is proposed that this solution also introduces a new YANG extension statement to indicate that a change is backwards-compatible.

When comparing a module schema, a tool would also be able to take into account enabled features, deviations, and the subset of the schema being used by the client. This would allow a tooling based approach to give a more accurate answer as to whether a client would be affected when upgrading between two software versions.

Solution advantages:

1. Gives the most accurate answer that works in all cases.

Solution disadvantages:

1. Cannot easily check whether two modules are compatible just by looking at them. Probably needs to be used in conjunction with a module level versioning scheme.

2. Differs from the scheme used by OpenConfig YANG models.

5.5. Follow existing RFC 7950 rules

The final choice is to decide that the existing mechanism described in RFC 7950, that disallows any non-backwards-compatible changes in a given model, is the best way forward. Instead of making a nbc chagne, the modeller can introduce new parallel nodes, and deprecate the existing nodes within the same module. Alternatively an entirely new module, with a separate name and namespace can be introduced.

As a solution, this cannot meet all of the requirements stated in the requirements draft.

If this solution was sufficient, then the YANG versioning design team would not have been formed. However, some vendors are pragmatically ignoring the strict YANG module update rules (e.g. for vendor modules).

Solution advantages:

1. No significant change in YANG language semantics required. Changes, or perhaps extensions, could be made to the YANG language to address some of other requirements that have independent solutions.

Solution disadvantages:

1. If an nbc has to be made (even for a minor feature) then there is a high impact to all clients using the module, servers implementing the module, and other YANG modules that import from the module. This impact would be particularly acute for a core YANG module that is being updated in an nbc way, that is imported by many other YANG modules. Hence, choosing this solution really means that there can be no nbc changes to a module unless the module is being restructured in a major way when a separate name for the module makes sense regardless.

2. Seems to make standardization slow because participants are seemingly try harder to get the perfect model first because the cost of having to change it seems so high.

3. Old, dead definitions can potentially never be removed from a module.

4. Does not work well for vendor generated YANG models, since they cannot easily have the level or control and stability required for it to never change.

5. Does not solve the problem where deviations are used to introduce nbc changes.

6. Introduces a problem where a single underlying property is represented by two (or more) independent data nodes in the same schema. There does not appear to be a clean solution on how to manage the relationship between these two nodes (e.g. if both an old and new client are interacting with a server). Other solutions have the potential of handling this better.

Variants:

One variant of this solution is to agree on the rules for making fixes to published YANG modules, and determine whether that requires any changes to the section 11 text in RFC 950.

6. Solutions to related YANG versioning issues

These partial solutions address particular point requirements. The partial solutions are:

1. Deprecated flag - Add a flag to YANG library to indicate whether deprecated nodes are implemented or not. This is a potential solution to Req 4.1.

2. Redefine deprecated stmt - Change the definition of the YANG deprecated statement to indicate that deprecated data nodes must be implemented, or otherwise deviated. This is a potential solution to Req 4.1.

3. Status description - Allow the "description" statement under the YANG "status" statement to document data node lifecycle, and allow for forward guidance. This is a potential solution to Reqs 4.2 and 4.3.

4. Alternative node path - Introduce a new YANG statement to provide an alternative path for a deprecated, or obsolete, data node. This is a potential additional solution to Req 4.2 and perhaps also Req 4.3.

7. Open Questions

This section lists some of the open questions that the design team is still grappling with.

7.1. Is YANG module revision date preserved?

With the introduction of the new versioning scheme, should every YANG module still have a revision statement, or is that entirely superseded by a new version statement? Is it required that YANG modules revision dates MUST be unique for different versions of a module?

The position that the DT is tending towards is:

> All revision dates for YANG modules must be unique. The slight complexity of requiring this should minimize the impact to existing tooling.
>
> it is acceptable to break the existing monotonically increasing property of the current module revision date, but within a given 'stream' of YANG modules the monotonically increasing property should be preserved.

7.2. Do YANG update rules allow for bug fixes?

Does YANG (RFC 7950) section 11 allow nbc fixes to existing models, and if so, are there any limits as to what form those fixes can take, or are these strictly prohibited by the module update rules?

7.3. Does one size fit all?

Potentially different types of YANG modules may want to follow different versioning semantics.

E.g. it may be right that standardized YANG modules are very slow changing and conservative in their backwards compatibility

Conversely, it is potentially more pragmatic that vendor YANG modules need to change in more significant ways mirroring changes in underlying implementations or hardware.

7.4. Should vendors we allowed to version YANG modules as part of a release train?

Some of the solutions described in this document probably require vendors to version vendor YANG modules outside of release trains, which is likely to be different to how some vendors are managing this today. Is it a reasonable constraint to put on vendors that they MUST version YANG modules outside of a release train to provide a cleaner version history?

7.5. How should versioning apply to submodules?

Submodules can have different revision dates from the including parent module. Does this mean that submodules should be versioned independently of their parent module? Or should the version number apply only at the module level?

Need to consider the upgrade rules allow definitions to be moved between submodules.

7.6. Is having a patch version number useful for YANG modules?

The semantic versioning solution on semver.org is designed to version both APIs and implementations. In this scenario, the patch level versioning number is particularly useful to indicate a fix in the implementation, where the API has not changed. The versioning for YANG modules is primarily concerned with the API semantics rather than implementation, and hence the patch level version number is not so directly useful, where its purpose is limited to changes that do not affect semantics of the YANG module (e.g. fixes to typos for example).

8. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The following people are members of that design team and have contributed to defining the problem and specifying the requirements:

- Balazs Lengyel
- Benoit Claise
- Ebben Aries
- Jason Sterne
- Joe Clarke
- Juergen Schoenwaelder
- Mahesh Jethanandani
- Michael (Wangzitao)
- Qin Wu
- Reshad Rahman

- Rob Wilton
- Susan Hares

9. Security Considerations

The document does not define any new protocol or data model. There is no security impact.

10. IANA Considerations

None

11. References

11.1. Normative References

[I-D.verdt-netmod-yang-versioning-reqs] Clarke, J., "YANG Module Versioning Requirements", draft-verdt-netmod-yang-versioning-reqs-01 (work in progress), October 2018.

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

[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>.

11.2. Informative References

[RFC8049] Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8049, DOI 10.17487/RFC8049, February 2017, <https://www.rfc-editor.org/info/rfc8049>.

[RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <https://www.rfc-editor.org/info/rfc8199>.

[RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <https://www.rfc-editor.org/info/rfc8299>.

Author's Address

Robert Wilton
Cisco Systems, Inc.

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 23, 2019

J. Clarke, Ed.
Cisco Systems, Inc.

October 20, 2018

YANG Module Versioning Requirements
draft-verdt-netmod-yang-versioning-reqs-01

Abstract

This document describes the problems that can arise because of the YANG language module update rules, that require all updates to YANG module preserve strict backwards compatibility. It also defines the requirements on any solution designed to solve the stated problems. This document does not consider possible solutions, nor endorse any particular solution.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 23, 2019.

Copyright Notice

Table of Contents

1. Introduction

This requirements document initially considers some of the existing YANG module update rules, then describes the problems that arise due to those rules embracing strict backwards compatibility, and finally defines requirements on any solution that may be designed to solve these problems by providing an alternative YANG versioning strategy.

2. Background

The YANG data modeling language [RFC7950] specifies strict rules for updating YANG modules (see section 11 "Updating a Module"). Citing a few of the relevant rules:

1. "As experience is gained with a module, it may be desirable to revise that module. However, changes to published modules are not allowed if they have any potential to cause interoperability problems between a client using an original specification and a server using an updated specification."

2. "Note that definitions contained in a module are available to be imported by any other module and are referenced in "import" statements via the module name. Thus, a module name MUST NOT be changed. Furthermore, the "namespace" statement MUST NOT be changed, since all XML elements are qualified by the namespace."

3. "Otherwise, if the semantics of any previous definition are changed (i.e., if a non-editorial change is made to any definition other than those specifically allowed above), then this MUST be achieved by a new definition with a new identifier."

4. "deprecated indicates an obsolete definition, but it permits new/ continued implementation in order to foster interoperability with older/existing implementations."

The rules described above, along with other similar rules, causes various problems, as described in the following sections:

2.1. Striving for model perfection

The points made above lead to the logical conclusion that the standardized YANG modules have to be perfect on day one (at least the structure and meaning), which in turn might explain why IETF YANG modules take so long to standardize. Shooting for perfection is obviously a noble goal, but if the perfect standard comes too late, it doesn't help the industry.

2.2. Some YANG Modules Are Not Backwards-Compatible

As we learn from our mistakes, we're going to face more and more non-backwards-compatible YANG modules. An example is the YANG data model for L3VPN service delivery [RFC8049], which, based on implementation experience, has been updated in a non-backwards-compatible way by [RFC8299].

While Standards Development Organization (SDO) YANG modules are obviously better for the industry, we must recognize that many YANG modules are actually generated YANG modules (for example, from internal databases), which is sometimes the case for vendor modules [RFC8199]. From time to time, the new YANG modules are not backwards-compatible.

Old module parts that are no longer needed, no longer supported, or are not used by consumers need to be removed from modules. It is often hard to decide which parts are no longer needed/used; still the need and practice of removing old parts exist. While it is rare in standard modules it is more common in vendor YANG modules where the usage of modules is more controlled.

The problems described in Section 2.7 may also result in incompatible changes.

In such cases, it would be better to indicate how backwards-compatible a given YANG module actually is.

As modules are sometimes updated in an incompatible way the current assumption that once a YANG module is defined all further revisions can be freely used as they are compatible is not valid.

2.3. Non-Backwards-Compatible Errors

Sometimes small errors force us to make non-backwards-compatible updates. As an example imagine that we have a string with a complex pattern (e.g., an IP address). Let's assume the initial pattern incorrectly allows IP addresses to start with 355. In the next version this is corrected to disallow addresses starting with 355. Formally this is a non-backwards-compatible change as the value space of the string is decreased. In reality an IP address and the implementation behind it was never capable of handling an address starting with 355. So practically this is a backwards-compatible change, just like a correction of the description statement. Current YANG rules are ambiguous as to whether non-backwards-compatible bug fixes are allowed without also requiring a module name change.

2.4. No way to easily decide whether a change is Backwards-Compatible

A management system, SDN controller, or any other user of a module should be capable of easily determining the compatibility between two module versions. Higher level logic for a network function, something that cannot be implemented in a purely model driven way, is always dependent on a specific version of the module. If the client finds that the module has been updated on the network node, it has to decide if it tries to handle it as it handled the previous version of the model or if it just stops, to avoid problems. To make this decision the client needs to know if the module was updated in a backwards-compatible way or not.

This is not possible to decide today because of the following:

- It is sometimes necessary to change the semantic behavior of a data node, action or rpc while the YANG definition does not change (with the possible exception of the description statement). In such a case it is impossible to determine whether the change is backwards-compatible just by looking at the YANG statements. It's only the human model designer who can decide.

- Problems with the deprecated and obsolete status statement, Section 2.7

- YANG module authors might decide to violate YANG 1.1 update rules for some of the reasons above.

Finding status changes or violations of update rules need a line-by-line comparison of the old and new modules is a tedious task.

2.5. No good way to specify which module revision to import

If a module (MOD-A) is imported by another one (MOD-B) the importer may specify which revision must be imported. Even if MOD-A is updated in a backwards-compatible way not all revisions will be suitable, e.g., a new MOD-B might need the newest MOD-A. However, both specifying or omitting the revision date for import leads to problems.

If the import by revision-date is specified

- If corrections are made to MOD-A these would not have any effect as the import's revision date would still point to the un-corrected earlier YANG module revision.

- If MOD-A is updated in a backwards-compatible way because another importer (MOD-C) needs some functionality, the new MOD-A could be used by MOD-B, but specifying the exact import revision-date prevents this. This will force the implementers to import two different revisions of MOD-A, forcing them to maintain old MOD-A revisions unnecessarily.

- If multiple modules import different revisions of MOD-A the human user will need to understand the subtle differences between the different revisions. Small differences would easily lead to operator mistakes as the operator will rarely check the documentation.

- Tooling/SW is often not prepared to handle multiple revisions of the same YANG module.

If the import revision-date is not specified

- any revision of MOD-A may be used including unsuitable ones. Older revisions may be lacking functionality MOD-B needs. Newer MOD-A revisions may obsolete definitions used by MOD-B in which case these must not be used by MOD-B anymore.

- As it is not specified which revisions of MOD-A are suitable for MOD-B. The problem has to be solved on a case by case basis studying all the details of MOD-A and MOD-B which is considerable work.

2.6. Early Warning about Removal

If a schema part is considered old/bad we need to be able to give advance warning that it will be removed. As this is an advance warning the part must still be present and usable in the current revision; however, it will be removed in one of the next revisions. The deprecated statement cannot be reliably used for this purpose both because deprecated nodes may not be implemented and also there is no mandate that text be provided explaining the deprecation.

We need the advance warning to allow users of the module time to plan/execute migration away from the deprecated functionality. Deprecation should be accompanied by information whether the functionality will just disappear or that there is an alternative, possibly more advanced solution that should be used.

Vendors use such warnings often, but the NMDA related redesign of IETF modules is also an example where it would be useful for IETF. As another example, see the usage of deprecated in the Java programming language.

2.7. Clear Indication of Node Support

The current definition of deprecated and obsolete in [RFC7950] (as quoted below) is problematic and should be corrected.

- "deprecated" indicates an obsolete definition, but it permits new/ continued implementation in order to foster interoperability with older/existing implementations.

- "obsolete" means that the definition is obsolete and SHOULD NOT be implemented and/or can be removed from implementations.

YANG is considered an interface contract between the server and the client. The current definitions of deprecated and obsolete mean that a schema node that is either deprecated or obsolete may or may not be implemented. The client has no way to find out which is the case except for by trying to write or read data at the leaf in question. This probing would need to be done for each separate data-node, which is not a trivial thing to do. This "may or may not" is unacceptable in a contract. In effect, this works as if there would be an if-feature statement on each deprecated schema node where the server

does not advertise whether the feature is supported or not. Why is it not advertised?

3. Terminology and Conventions

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

In addition, this document uses the following terminology:

- YANG module revision: An instance of a YANG module, with no implied ordering or backwards compatibility between different revisions of the same module."

- YANG module version: A YANG module revision, but also with an implied partial ordering relationship between other versions of the same module. Each module version must be uniquely identifiable.

- Non-backwards-compatible (NBC): In the context of this document, the term 'non-backwards-compatible' refers to a change or set of changes between two YANG module revisions that do not adhere to the list of allowable changes specified in Section 11 "Updating a Module" of [RFC7950], with the following additional clarification:

  * Any addition of, or change to, a "status" statement that allows a server to remove support for a schema node is considered a non-backwards-compatible change

4. The Problem Statement

Considering the issues described in the background, the problem definition can be summarized as follows.

Development of data models for a large collection of communication protocols and system components is difficult and typically only manageable with an iterative development process. Agile development approaches advocate evolutionary development, early delivery, and continual improvement. They are designed to support rapid and flexible response to change. Agile development has been found to be very successful in a world where the objects being modeled undergo constant changes.

The current module versioning scheme relies on the fundamental idea that a definition, once published, never changes its semantics. As a consequence, if a new definition is needed with different non-backwards-compatible semantics, then a new definition must be created

to replace the old definition. The advantage of this versioning scheme is that a definition identified by a module name and a path has fixed semantics that never change. (The details are a bit more nuanced but we simplify things here a bit in order to get the problems worked out clearly.)

There are two main disadvantages of the current YANG versioning scheme:

- Any non-backwards-compatible change of a definition requires either a new module name or a new path. This has been found costly to support in implementations, in particular on the client side.

- Since non-backwards-compatible changes require either a new module name or a new path, such changes will impact other modules that import definitions. In fact, with the current module versioning scheme other modules have to opt-in in order to use the new version. This essentially leads to a ripple effect where a non-backwards-compatible change of a core module causes updates on a potentially large number of dependent modules.

Other problems experienced with the current YANG versioning scheme are the following:

- YANG has a mechanism to mark definitions deprecated but it leaves it open whether implementations are expected to implement deprecated definitions and there is no way (other than trial and error) for a client to find out whether deprecated definitions are supported by a given implementation.

- YANG does not have a robust mechanism to document which data definitions have changed and to provide guidance how implementations should deal with the change. While it is possible to have this described in general description statements, having these details embedded in general description statements does not make this information accessible to tools.

- YANG data models often do not exist in isolation and they interact with other software systems or data models that often do allow (controlled) non-backwards-compatible changes. In some cases, YANG models are mechanically derived from other data models that do allow (controlled) non-backwards-compatible changes. In such situations, a robust mapping to YANG requires to have version numbers exposed as part of the module name or a path definition, which has been found to be expensive on the client side (see above).

Given the need to support agile development processes and the disadvantages and problems of the current YANG versioning scheme described above, it is necessary to develop requirements and solutions for a future YANG versioning scheme that better supports agile development processes, whilst retaining the ability for servers to handle clients using older versions of YANG modules.

5. Requirements of a YANG Versioning Solution

The following is a list of requirements that a solution to the problems mentioned above MUST or SHOULD have. The list is grouped by similar requirements but is not presented in a set priority order.

1. Requirements related to making non-backwards-compatible updates to modules:

    1.1 A mechanism is REQUIRED to update a module in a non-backwards-compatible way without forcing all modules with import dependencies on the updated module from being updated at the same time (e.g. to change its import to use a new module name).

    1.2 Non-backwards-compatible updates of a module MUST not impact clients that only access data nodes of the module that have either not been updated or have been updated in backwards-compatible ways.

    1.3 A refined form of YANG's 'import' statement MUST be provided that is more restrictive than "import any revision" and less restrictive than "import a specific revision". Once non-backwards-compatible changes to modules are allowed, the refined import statement is used to express the correct dependency between modules.

    1.4 The solution MUST allow for modules to be versioned by software release. In particular, backwards-compatible enhancements and bug fixes MUST be allowed in any non-latest release.

2. Requirements related to identifying changes between different module revisions:

    2.1 Readers of modules, and tools that use modules, MUST be able to determine whether changes between two revisions of a module constitute a backwards-compatible or non-backwards-compatible version change. In addition, it MAY be helpful to identify whether changes represent bug fixes, new functionality, or both.

2.2 A mechanism SHOULD be defined to determine whether data nodes between two arbitrary YANG module revisions have (i) not changed, (ii) changed in a backwards-compatible way, (iii) changed in a non-backwards-compatible way.

3. Requirements related to supporting existing clients in a backwards-compatible way:

3.1 The solution MUST provide a mechanism to allow servers to support existing clients in a backwards-compatible way.

3.2 The solution MUST provide a mechanism to allow servers to simultaneously support clients using different revisions of modules. A client's choice of particular revision of one or more modules may restrict the particular revision of other modules that may be used in the same request or session.

3.3 Clients are expected to be able to handle unexpected instance data resulting from backwards-compatible changes.

4. Requirements related to managing and documenting the life cycle of data nodes:

4.1 A mechanism is REQUIRED to allow a client to determine whether deprecated nodes are implemented by the server.

4.2 If a data node is deprecated or obsolete then it MUST be possible to document in the YANG module what alternatives exist, the reason for the status change, or any other status related information.

4.3 A mechanism is REQUIRED to indicate that certain definitions in a YANG module will become status obsolete in future revisions but definitions marked as such MUST still be implemented by compliant servers.

4.4 If multiple revisions of a YANG module are published, then the solution SHOULD allow for bug fixes to be made to an older revision of the module.

5. Requirements related to documentation and education:

5.1 The solution MUST provide guidance to model authors and clients on how to use the new YANG versioning scheme.

5.2 The solution is REQUIRED to describe how to transition from the existing YANG 1.0/1.1 versioning scheme to the new scheme.

5.3 The solution MUST describe how the versioning scheme affects the interpretation of instance data and references to instance data, for which the schema definition has been updated in a non-backwards-compatible way.

6. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The following people are members of that design team and have contributed to defining the problem and specifying the requirements:

- Balazs Lengyel
- Benoit Claise
- Ebben Aries
- Jason Sterne
- Joe Clarke
- Juergen Schoenwaelder
- Mahesh Jethanandani
- Michael (Wangzitao)
- Qin Wu
- Reshad Rahman
- Rob Wilton

7. Acknowledgments

The design team would like to thank Christian Hopps and Vladimir Vassilev for their feedback and perspectives in shaping and fine tuning the versioning requirements.

8. Security Considerations

The document does not define any new protocol or data model. There is no security impact.

9. IANA Considerations

None

10. References

10.1. Normative References

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

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

10.2. Informative References

[RFC8049] Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8049, DOI 10.17487/RFC8049, February 2017, <https://www.rfc-editor.org/info/rfc8049>.

[RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <https://www.rfc-editor.org/info/rfc8199>.

[RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <https://www.rfc-editor.org/info/rfc8299>.

Author's Address

Joe Clarke (editor)
Cisco Systems, Inc.
7200-12 Kit Creek Rd
Research Triangle Park, North Carolina
United States of America

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

NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 18, 2019

Q. Wu
R. Ranade
Huawei

September 14, 2018

NMDA Base Notification for Applied Intended Configuration
draft-wu-netmod-base-notification-nmda-00

Abstract

The Network Configuration Protocol (NETCONF)and RESTCONF provides mechanisms to manipulate configuration datastores. NMDA introduces additional datastores for systems that support more advanced processing chains converting configuration to operational state. However, client applications are not able to be aware of common events in these additional datstores of the management system, such as a applied configuration state change in NETCONF server or RESTCONF server, that may impact management applications. This document define a YANG module that allows a client to receive additional notifications for some common system events pertaining to the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

Status of This Memo

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

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

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

This Internet-Draft will expire on March 18, 2019.

Table of Contents

1. Introduction

The Network Configuration Protocol (NETCONF) [RFC6241] and RESTCONF [RFC8040] provides mechanisms to manipulate configuration datastores. NMDA introduces additional datastores (e.g., <intended>, <operational>) for systems that support more advanced processing chains converting configuration to operational state. However, client applications are not able to be aware of common events in these additional datastores of the management system, e.g., there are many background activities (e.g.,NMDA datastore consistency checking [NMDA-DIFF]) that happen during the time that configuration is committed to <running> to the time that the configuration is actually applied to <operational>. It is possible that some configuration could not be applied to <operational> due to either validation issues, or missing resource etc. There is a need for user to know the applying result of <intended> data-store and the reason why the configuration were not applied.

This document define a YANG module that allows a NMS client to receive additional notifications for some common system events pertaining to the Network Management Datastore Architecture (NMDA) defined in [RFC8342]. These notifications are designed to support the monitoring of the base system events within the server and not specific to any network management protocols such as NETCONF and RESTCONF.

The solution presented in this document is backwards compatible with [RFC6470].

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 [RFC8342] and are not redefined here:

- operational state datastore
- running configuration datastore
- intended configuration datastore

2. NMDA Base Notifications for applied intended configuration

2.1. Overview

The YANG module in NETCONF Base Notifications [RFC6470] specifies the following 5 event notifications for the 'NETCONF' stream to notify a client application that the NETCONF server state has changed:

- netconf-config-change
- netconf-capability-change
- netconf-session-start
- netconf-session-end
- netconf-confirmed-commit

These event notifications used within the 'NETCONF' stream are accessible to clients via the subscription mechanism described in [RFC5277].

This document introduces NMDA specific extension which allows a client to receive 3 notifications for additional common system events as follows:

intended-apply-start: Generated when a server with network management protocol support detects that configuration applied

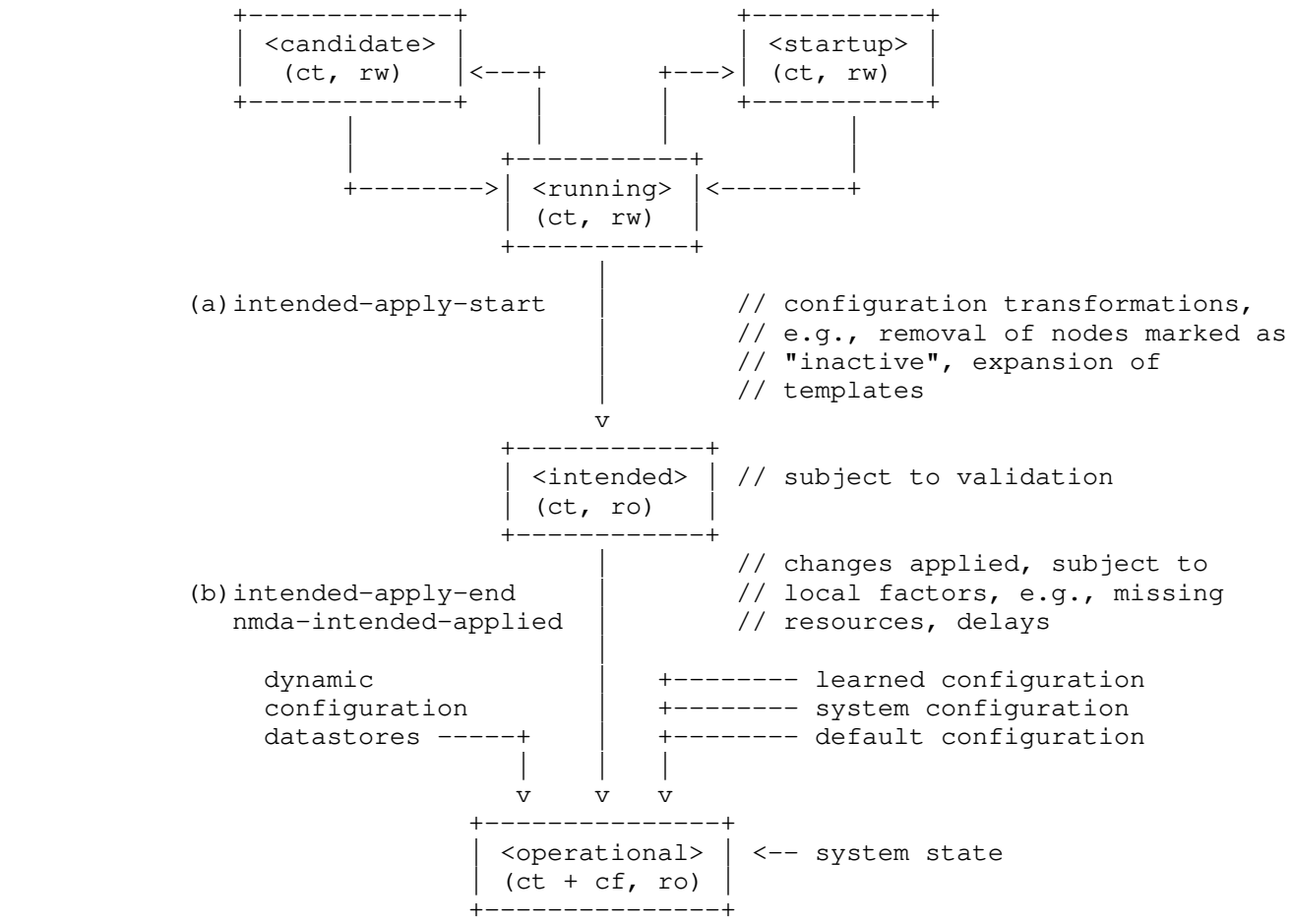
from intended has started. The applied-intended-start starts at the time when configuration is succesfully committed to <running>, i.e., the confirmed-commit procedure has been completed.A server MAY optionally generate this event for both NETCONF session and non-NETCONF management sessions. In case two commits are detected, if the second commit is confirmed commit, the commit and confirmed commit should be treated as the same commit, i.e., intended-apply-start commit should not be sent for persistent confirmed commit. if the the second commit is not confirmed commit, intended-apply-start for the second commit should not be sent until intended-apply-end or nmda-intended-applied for the first commit has been sent.

intended-apply-end: Generated when a server with network management protocol support detects that all or a set of configurations are successfully applied or none of them are applied. Note that configuration applying process of the learned configuration, system-provided configuration, and default values defined by data models is not relevant to the events defined in this document. Unlike the learned configuration, system-provided configuration, the intended configuration is not applied frequently and may be fixed after the configuration applying process. A server MAY optionally generate this event for both NETCONF session and non-NETCONF management sessions.

nmda-intended-applied: Generated when a server with network management protocol support detects that all or a set of configurations are successfully applied or none of them are applied. Occurs at the same as intended-apply-end and Indicates the event and the current state of the intended configuration applying. If the applied-event of the mmda-intended-applied is set to start or compete, the nmda-intended-applied can also used to indicate the start time and end time of the intended configuration applying procedure, i.e., intended-apply-start and intended-apply-end are not needed being sent to the client. In addition, NMDA datastore compare [NMDA-DIFF] should be used to check which part of intended configuration data is applied or which part of intended configuration data is not applied. A server MAY report events for non- NETCONF management sessions (such as RESTCONF,gPRC), using the 'session-id' value of zero.

Editor-Note: If nmda-intended-applied can be used to indicate start time and end time of the intended configuration applying procedure, do we need intended-apply-start and intended-apply-end events?

The following figure shows event notification sequence defined in this document.



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

These notification messages are accessible to clients via either the subscription mechanism described in [RFC5277] or dynamic subscription mechanism and configured subscription mechanism described in [I-D.ietf-netconf-netconf-event-notifications].

2.2. Data Model Design

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

```
   module: ietf-nmda-notifications
     notifications:
       +---n intended-apply-start
       |  +--ro username           string
       |  +--ro session-id         session-id-or-zero-type
       |  +--ro source-host?       inet:ip-address
       |  +--ro commit-persist-id  string
       +---n intended-apply-end
       |  +--ro username           string
       |  +--ro session-id         session-id-or-zero-type
       |  +--ro source-host?       inet:ip-address
       |  +--ro commit-persist-id  string
       |  +--ro (complete-status)?
       |     +--:(global-errors)
       |     |  +--ro errors
       |     |     +--ro error*
       |     |        +--ro error-type       enumeration
       |     |        +--ro error-tag        string
       |     |        +--ro error-app-tag?   string
       |     |        +--ro error-path?      instance-identifier
       |     |        +--ro error-message?   string
       |     |        +--ro error-info?
       |     +--:(ok)
       |        +--ro ok?            empty
       +---n nmda-intended-applied
          +--ro username               string
          +--ro session-id             session-id-or-zero-type
          +--ro source-host?           inet:ip-address
       |  +--ro commit-persist-id      string
          +--ro applied-event          enumeration
          +--ro (applied-status)?
          |  +--:(global-errors)
          |  |  +--ro errors
          |  |     +--ro error*
          |  |        +--ro error-type       enumeration
          |  |        +--ro error-tag        string
          |  |        +--ro error-app-tag?   string
          |  |        +--ro error-path?      instance-identifier
          |  |        +--ro error-message?   string
          |  |        +--ro error-info?
          |  +--:(ok)
          |     +--ro ok?                    empty
          +--ro fail-applied-target*
             +--ro datastore?   identityref
             +--ro edit-id?     string
             +--ro target?      ypatch:target-resource-offset
             +--ro (applied-status-choice)?
                +--:(errors)
```

```
                  +--ro errors
                     +--ro error*
                        +--ro error-type       enumeration
                        +--ro error-tag        string
                        +--ro error-app-tag?   string
                        +--ro error-path?      instance-identifier
                        +--ro error-message?   string
                        +--ro error-info?
```

The following are examples of a nmda-intended-applied notification message:

```
  <notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2017-06-16T16:30:59.137045+09:00</eventTime>
    <nmda-intended-applied xmlns="urn:ietf:params:xml:ns:yang:ietf-nmda-notifica
tions">
      <username>admin</username>
      <session-id>0</session-id>
      <source-host>10.251.93.83</source-host>
      <commit-persist-id>IQ,d4668</commit-persist-id>
      <applied-event>complete<applied-event>
      <errors>
        <error-type>protocol</error-type>
        <error-tag>mis-resource</error-tag>
        <error-path xmlns:ops="https://example.com/ns/ietf-interfaces">\
         \/if:interfaces-state\
        \</error-path>
        <error-message>refer to resources that are not \
        \available or otherwise not physically present.\
        \</error-message>
      </errors>
      <fail-applied-taget>
        <datastore>intended</datastore>
        <edit-id>1<edit-id>
        <target>/ietf-interfaces:interfaces-state</target>
      </fail-applied-target>
      <fail-applied-taget>
        <datastore>intended</datastore>
        <edit-id>1<edit-id>
        <target>/ietf-system:system</target>
      </fail-applied-target>
   </nmda-intended-applied>
  </notification>
```

2.3. Relation with NMDA Datastore Compare

NMDA datastore compare [NMDA-DIFF]could be used to check which part of intended configuration data is applied or which part of intended configuration data is not applied. On the other hand, the event

notification for applied-intended-start can be used to trigger NMDA datastore compare procedure to be started.

2.4. Definitions

This section presents the YANG module defined in this document. This module imports data types from the 'ietf-datastores' module defined in [RFC8342] and 'ietf-inet-types' module defined in [RFC6021].

```
 <CODE BEGINS> file "ietf-nmda-notifications@2018-04-01.yang"
module ietf-nmda-notifications {
  namespace "urn:ietf:params:xml:ns:yang:ietf-nmda-notifications";
  prefix ndn;
  import ietf-datastores {
    prefix ds;
  }
  import ietf-inet-types { prefix inet; }
  import ietf-yang-patch {
    prefix ypatch;
  }
  import ietf-restconf { prefix rc;}
  organization
    "IETF NETMOD (Network Modeling) Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>
     WG Chair: Kent Watsen
               <mailto:kwatsen@juniper.net>
     Editor:   Qin Wu
               <mailto:bill.wu@huawei.com>
     Editor:   Rohit R Ranade
               <mailto:rohitrranade@huawei.com>";
  description
    "This module defines a YANG data model for use with the
     NETCONF and RESTCONF protocol that allows the client to
     receive additional common event notifications related to NMDA.
     Copyright (c) 2012 IETF Trust and the persons identified as
     the document authors.  All rights reserved.
     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).
     This version of this YANG module is part of RFC xxxx; see
     the RFC itself for full legal notices.";

  revision 2018-04-01 {
```

```
    description
      "Initial version.";
    reference "RFC xxx: NETCONF Base Notifications for NMDA";
  }
  typedef session-id-or-zero-type {
     type uint32;
     description
       "NETCONF Session Id or Zero to indicate none";
   }

  grouping common-session-parms {
    description
      "Common session parameters to identify a
       management session.";

    leaf username {
      type string;
      mandatory true;
      description
        "Name of the user for the session.";
    }

    leaf session-id {
      type session-id-or-zero-type;
      mandatory true;
      description
        "Identifier of the session.
         A NETCONF session MUST be identified by a non-zero value.
         A non-NETCONF session MAY be identified by the value zero.";
    }

    leaf source-host {
      type inet:ip-address;
      description
        "Address of the remote host for the session.";
    }

   leaf commit-persist-id {
     type string;
     description
       "This parameter is used to identify each commit.
        The value of this parameter is a token that must be given
        in the 'persist-id' parameter of <commit> or
        <cancel-commit> operations in order to confirm or cancel
        the persistent confirmed commit.

        The token should be a random string.";
       reference "RFC 6241, Section 8.3.4.1";
```

```
   }
  }

   notification intended-apply-start {
       description
         " Generated when a server detects that applied configuration from
           intended has started. This event will be sent immediately upon
           any data is written to <running>. A server MAY generate this
           event for both NETCONF session and non-NETCONF management
           sessions.";
       uses common-session-parms;
     } // notification applied-intended-start

   notification intended-apply-end {
       description
         " Generated when a server detects that a applied configuration
          from intended has complete.  A server MAY optionally generate
          this event for both NETCONF session and non-NETCONF management
          sessions.";
       uses common-session-parms;
      choice complete-status {
           description
             "Report global errors or complete success.
              If there is no case selected, then errors
              are reported in the 'edit-status' container.";
           case global-errors {
             uses rc:errors;
             description
               "This container will be present if global errors that
                are unrelated to a specific edit occurred.";
           }
           leaf ok {
             type empty;
             description
               "This leaf will be present if the request succeeded
                and there are no errors reported in the 'edit-status'
                container.";
           }
         }
     } // notification applied-intended-complete

    notification nmda-intended-applied {
      description
        "Generated when a server detects that a
         intended configuration applied event has occurred. Indicates
         the event and the current state of the intended data applying
          procedure in progress.";
      reference "RFC 8342, Section 5";
```

```
      uses common-session-parms;
      leaf applied-event {
        type enumeration {
          enum "start" {
            description
              "The intended data applying procedure has started.";
           }
          enum "transform" {
              description
              " <intended> MAY be updated independently of <running>
                due to configuration transformation.";
          }
          enum "synchronizing" {
              description
              "The data validation results within <intended> is in
               progress of synchronizing to <operatonal>.";
          }
           enum "complete" {
             description
               "The intended data applying procedure has been
                completed.";
           }
        }
        mandatory true;
        description
          "Indicates the event that caused the notification.";
      }
     choice applied-status {
           when "./applied-event = 'complete'";
           description
             "Report global errors or complete success.
              If there is no case selected, then errors
              are reported in the 'edit-status' container.";
           case global-errors {
             uses rc:errors;
             description
               "This container will be present if global errors that
                are unrelated to a specific edit occurred.";
           }
          leaf ok {
            type empty;
            description
              "This leaf will be present if the request succeeded
               and there are no errors reported in the 'edit-status'
               container.";
           }
         }
```

```
      list fail-applied-target {
        leaf datastore {
          type identityref {
            base ds:datastore;
          }
          default "ds:operational";
          description
            "Indicates which datastore has changed or which datastore is
             target of edit-data operation.";
        }
        leaf edit-id {
               type string;
                description
                  "Response status is for the 'edit' list entry
                   with this 'edit-id' value.";
              }
        leaf target {
          type ypatch:target-resource-offset;
          description
            "Topmost node associated with the configuration change.
             A server SHOULD set this object to the node within
             the datastore that is being altered.  A server MAY
             set this object to one of the ancestors of the actual
             node that was changed, or omit this object, if the
             exact node is not known.";
        }
        choice applied-status-choice {
               description
                 "A choice between different types of status
                  responses for each 'edit' entry.";
               case errors {
                 uses rc:errors;
                 description
                   "The server detected errors associated with the
                    edit identified by the same 'edit-id' value.";
               }
              }
        description
          "List for fail applied targets";
      }
    }
  }
```

<CODE ENDS>

3. 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, defined in [RFC6242].

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:

/applied-intended-start:

> Event type itself indicates that a server may start applying configuration from intended. It may be possible for an attacker to slow down intended validation or update intended independently of running by somehow taking advantage of configuration template transformation.

/applied-intended-complete:

> Event type itself indicates that a server may be finished applying configuration from intended. This event could alert an attacker that a datastore may have been altered.

/nmda-data-applied/applied-event:

> Indicates the specific applied intended applied event state change that occurred. A value of 'complete' probably indicates that intended data applying procedure has completed.

4. IANA Considerations

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

This document registers one module name in the 'YANG Module Names' registry, defined in [RFC7950]:

> name: ietf-nmda-notifications

prefix: ndn

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

RFC: xxxx

5. Acknowledgements

Thanks to Juergen Schoenwaelder, Alex Clemm,Carey Timothy and Andy Berman to review this draft and Thank Xiaojian Ding provide important input to the initial version of this document.

6. Normative References

[I-D.clemm-netmod-nmda-diff]
Clemm, A., Qu, Y., Tantsura, J., and A. Bierman, "Comparison of NMDA datastores", draft-clemm-netmod-nmda-diff-00 (work in progress), June 2018.

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

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

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <https://www.rfc-editor.org/info/rfc5277>.

[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>.

[RFC6021] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6021, DOI 10.17487/RFC6021, October 2010, <https://www.rfc-editor.org/info/rfc6021>.

[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>.

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

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

[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>.

Authors' Addresses

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

Email: bill.wu@huawei.com

Rohit R Ranade
Huawei
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560066
India

Email: rohitrranade@huawei.com

NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 22, 2019

Q. Wu
Huawei
B. Lengyel
Ericsson Hungary
Y. Niu
Huawei

October 19, 2018

Factory default Setting
draft-wu-netmod-factory-default-01

Abstract

This document defines a method to reset a YANG datastore to its factory-default content. The reset operation may be used e.g. during initial zero-touch configuration or when the existing configuration has major errors, so re-starting the configuration process from scratch is the best option.

A new reset-datastore RPC is defined. Several methods of documenting the factory-default content are specified.

Optionally a new "factory-default-running" read-only datastore is defined, that contains the data that will be copied over to the running datastore at reset.

Table of Contents

1. Introduction

This document defines a method to reset a YANG datastore to its factory-default content. The reset operation may be used e.g. during initial zero-touch configuration or when the existing configuration has major errors, so re-starting the configuration process from scratch is the best option. When resetting a datastore all previous configuration settings will be lost and replaced by the factory-default content.

A new reset-datastore RPC is defined. Several methods of documenting the factory-default content are specified.

Optionally a new "factory-default-running" read-only datastore is defined, that contains the data that will be copied over to the running datastore at reset. This datastore can also be used in <get-data> operation.

NETCONF defines the <delete> operation that allows resetting the <startup> datastore and the <discard-changes> operation that copies the content of the <running> datastore into the <candidate> datastore. However it is not possible to reset the running datastore, to reset the candidate datastore without changing the running datastore or to reset any dynamic datastore.

A RESTCONF server MAY implement the above NETCONF operations, but that would still not allow it to reset the running configuration.

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 [RFC8342] and are not redefined here:

- startup configuration datastore
- candiate configuration datastore
- running configuration datastore
- intended configuration datastore
- operational state datastore

The following terms are defined in this document as follows:

- factory-default datastore: A read-only datastore holding a preconfigured minimal initial configuration that can be used to initialize the configuration of a server. The content of the datastore is usually static, but MAY depend on external factors like available HW.

2. Reset-Datastore RPC

A new "reset-datastore" RPC is introduced. It will have a target datastore as a parameter. Upon receiveing the RPC the YANG server resets the content of the target datastore to its factory-default content. Only writable datastores can be specified as a target. Read-only datastores receive their content from other datastores (e.g. <intended> gets its content from <running>).

Factory-default content SHALL be specified by one of the following means in order of precedence

1. For the <running>, <candidate> and <startup> datastores as the content of the <factory-default> datastore, if it exists

2. YANG Instance Data [I-D.lengyel-netmod-yang-instance-data]

3. In some implementation specific manner

4. For dynamic datastores unless otherwise specified the factory-default content is empty.

3. Factory-Default Datastore

This document introduces a new datastore resource named 'Factory-Default' that represents a preconfigured minimal initial configuration that can be used to initialize the configuration of a server.

- Name: "factory-default"

- YANG modules: all

- YANG nodes: all "config true" data nodes

- Management operations: The content of the datastore is set by the YANG server in an implementation dependent manner. The content can not be changed by management operations via NETCONF, RESTCONF, the CLI etc. unless specialized, dedicated operations are provided. The contents of the datastore can be read using NETCONF, RESTCONF <get-data> operation. The operation <reset-datastore> can be used to copy the content of the datastore to another datastore. The content of the datastore is not propagated auomatically to any other datastores.

- Origin: This document does not define a new origin identity as it does not interact with <operational> datastore.

- Protocols: RESTCONF, NETCONF and other management protocol.

- Defining YANG module: "ietf-factory-default"

The datastore content is usually defined by the device vendor. It is usually static, but MAY change e.g. depending on external factors like HW available or during device upgrade.

On devices that support non-volatile storage, the contents of <factory > MUST persist across restarts

4. YANG Module

```
<CODE BEGINS> file "ietf-factory-default.yang"
module ietf-factory-default {
  yang-version 1.1;
  namespace urn:ietf:params:xml:ns:yang:ietf-factory-default;
  prefix fdef;

  import ietf-netconf { prefix nc ; }
  import ietf-datastores { prefix ds; }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web:   <https://tools.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>

     WG Chair: Kent Watsen
               <mailto:kwatsen@juniper.net>

     WG Chair: Mahesh Jethanandani
               <mailto:mjethanandani@gmail.com>

     Editor:   Balazs Lengyel
               <mailto:balazs.lengyel@ericsson.com>

     Editor:   Qin Wu
               <mailto:bill.wu@huawei.com>";

  description
    "This module defines the
    - reset-datastore RPC
    - factory-default datastore
    - an extension to the Netconf <copy-config> operation to
    allow it to operate on the factory-default datastore.

    It provides functionality to reset a YANG datastore to its
```

```
    factory-default content.

     Copyright (c) 2018 IETF Trust and the persons identified as
     authors of the code. All rights reserved.

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

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
     NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and
     'OPTIONAL' in the module text are to be interpreted as described
     in RFC 2119 (https://tools.ietf.org/html/rfc2119).

     This version of this YANG module is part of RFC XXXX
     (https://tools.ietf.org/html/rfcXXXX); see the RFC itself for
     full legal notices.";

  revision 2018-10-09 {
    description
      "Initial revision.";
      reference "RFC XXXX: Factory default Setting Capability for
      RESTCONF";
  }

  feature factory-default-as-datastore {
    description "Indicates that the factory default configuration is
      also available as a separate datastore";
  }

  rpc reset-datastore {
    description "The target datastore is reset to its factory
      default content. ";

    input {
      leaf-list target-datasore {
        type identityref {
          base "ds:datastore" ;
        }
        min-elements 1;
        description "The datastore(s) whose content will be
          replaced by the factory-default configuration.";
      }
      // Do we need an extra parameter that may order a restart of
      // the YANG-server or the whole system?
```

```
    }
  }

  identity factory-default {
    if-feature factory-default-as-datastore;
    base ds:datastore;
    description "The read-only datastore contains the configuration that
      will be copied into e.g. the running datastore by the
      reset-datastore operation if the target is the running
      datastore.";
  }
}
<CODE ENDS>
```

5. IANA Considerations

This document registers one URI in the IETF XML Registry [RFC3688]. The following registration has been made:

URI: urn:ietf:params:xml:ns:yang:ietf-factory-default

Registrant Contact: The IESG.

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

This document registers one YANG module in the YANG Module Names Registry [RFC6020]. The following registration has been made:

name: ietf-factory-default

namespace: urn:ietf:params:xml:ns:yang:ietf-factory-default

prefix: fdef

RFC: xxxx

6. Security Considerations

The <reset-datastore> RPC can overwrite important and security sensitive information in one of the other datastores e.g. running, therefore it is important to restrict access to this RPC using the standard access control methods. [RFC8341]

The content of the factory-default datastore is usually not security sensitive as it is the same on any device of a certain type.

7. Acknowledgements

Thanks to Juergen Schoenwaelder, Ladislav Lhotka to review this draft and provide important input to this document.

8. Contributors

Rohit R Ranade
Huawei
Email: rohitrranade@huawei.com

9. References

9.1. Normative References

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

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

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

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

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

9.2. Informative References

[I-D.ietf-netconf-zerotouch]
Watsen, K., Abrahamsson, M., and I. Farrer, "Zero Touch Provisioning for Networking Devices", draft-ietf-netconf-zerotouch-25 (work in progress), September 2018.

[I-D.lengyel-netmod-yang-instance-data]
Lengyel, B. and B. Claise, "YANG Instance Data Files and their use for Documenting Server Capabilities", draft-lengyel-netmod-yang-instance-data-04 (work in progress), October 2018.

Appendix A. Open Issues

- Do we need a restart after <reset-datastore> ? What kind of restart, just the YANG-Server or the full system?

- Do we need the concept of reboot? How is that different from a restart? Does it result in some sort of reset-datastore?

Appendix B. Difference between <startup> datastore and <factory-default> datastore

When the device first boots up, the content of the <startup> and <factory-default> will be identical. The content of <startup> can be subsequently changed by using <startup> as a target in a <copy-config> operation. The <factory-default> is a read-only datastore and it is usually static as described in earlier sections.

Appendix C. Changes between revisions

v3 - v00 - v01

- Changed name from draft-wu-netconf-restconf-factory-restore to draft-wu-netmod-factory-default

- Removed copy-config ; reset-datastore is enough

v02 - v03

- Restructured

- Made new datastore optional

- Removed Netconf capability

- Listed Open issues

v01 - v02

- -

v00 - v01

o -

Authors' Addresses

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

Email: bill.wu@huawei.com

Balazs Lengyel
Ericsson Hungary
Magyar Tudosok korutja 11
1117 Budapest
Hungary

Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

Ye Niu
Huawei

Email: niuye@huawei.com

NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 3, 2019

M. Wang
Q. Wu
Huawei
C. Xie
China Telecom

September 30, 2018

A YANG Data model for Event Management
draft-wwx-netmod-event-yang-00

Abstract

This document defines an YANG data model for event management [RFC7950]. The Event YANG provides the ability to monitor state change on the local system or on a remote system and take simple action when a trigger condition on system state is met.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 3, 2019.

Table of Contents

1. Introduction

This document defines an Event YANG data model [RFC7950]. The Event YANG provides the ability to monitor state changes on the local system or on a remote system and take simple action when a trigger condition on system state is met.

The data model in this document is designed to be compliant with the Network Management Datastore Architecture (NMDA) [RFC8342].

2. Conventions used in this document

2.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying [RFC2119] significance.

This document uses the following terms:

Error A deviation of a system from normal operation [RFC3877].

Fault Lasting error or warning condition [RFC3877].

Event Something that happens which may be of interest or trigger the invocation of the rule. A fault, an alarm, a change in network state, network security threat, hardware malfunction, buffer untilization crossing a threshold, network connection setup, an external input to the system, for example [RFC3877].

2.2. Tree Diagrams

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

3. Objectives

This section describes some of the design objectives for the Event Data Model:

The Event YANG should provide the ability to monitor yang instance on a local or remote system using the NETCONF/RESTCONF, and initiates simple actions whenever a trigger condition is met. For example, a NETCONF subscribed notification can be generated when an instance value exceeds the threshold.

Clear and precise identification of Event types and instances.

Allow the server to inform the client that certain Events are related to other Events.

Allow one event to be able to trigger another external event or generate derived events.

The event data model defined in this document can be implemented on a system that also implements EVENT-MIB; thus, the mapping between the event data model and ENTITY-MIB should be clear.

4. Relationship to YANG PUSH

YANG-push mechanism provides a subscription service for updates from a datastore. And it support two types of subscriptions which are distinguished by how updates are triggered: periodic and on-change.

The On-change PUSH allow receivers to receive updates whenever changes to targeted objects occur. This document specifies a mechanism that provides three trigger conditions:

- Existence: When a specific yang instance appears, the trigger fires. E.g. Reserved ports are configured.

- Boolean: If the test result is true the trigger fires. The trigger will not fire again until the value has become false and come back to true. And the user can set the type of boolean comparison (e.g. unequal, equal, less, less-or-equal, greater, greater-or-equal, etc), so for example if the boolean comparison type is 'less' the trigger will be fired if the value of YANG Instance is less than the value of Boolean Value.

- Threshold: The event that may be triggered when a YANG instance at data-instance list is found. If the first sample after this instance becomes active is greater than or equal to 'rising-value' and the 'startup' is equal to 'rising' then one threshold rising event is triggered for that instance. And if the first sample after this instance becomes active is less than or equal to 'falling-value' and the 'startup' is equal to 'falling' then one threshold falling event is triggered for that instance.

And the YANG PUSH mechanism more focus on the remote mirroring and monitoring of configuration and operational state. For example, for on change method, the subscriber will receive a notification if the changes appears. The model defines in this document provides a method which allow automatic setting the value of the corresponding instance node when some event is triggered. It esbalishes connection between network service monitoring and network service provision and can use output generated by network service monitoring as input of network service provision and thereby provide automated network management. The details of the usage example is described in Appendix A.

5. Relationship to EVENT MIB

If the device implements the EVENT-MIB [RFC2981], each entry in the "/events/event/trigger" list is mapped to MteTriggerEntry,MteTriggerExistenceEntry,MteTriggerBooleanEntry,MteTriggerThresholdEntry,MteObjectsEntry,MteEventEntry,MteEventSetEntry. respectively.

The following table lists the YANG data nodes with corresponding objects in the EVENT-MIB [RFC2981].

YANG data node in ietf-event.yang	EVENT-MIB Objects (RFC2981)
evt-smp-min	mteResourceSampleMinimum
evt-smp-instance-max	mteResourceSampleInstanceMaximum
traget	mteObjectsName
event-name	mteEventName
event-description	mteEventComment
value	mteEventSetValue
events/event/trigger/name	mteTriggerName
trigger-description	mteTriggerComment
frequency	mteTriggerFrequency
comparison	mteTriggerBooleanComparison
value	mteTriggerBooleanValue
rising-event	mteTriggerThresholdRising
falling-event	mteTriggerThresholdFalling
delta-rising-event	mteTriggerThresholdDeltaRising
threshold/startup	mteTriggerThresholdStartup
existence/enable	mteTriggerExistenceStartup
boolean/enable	mteTriggerBooleanStartup

6. Model Overview

The event yang has four lists: trigger, target, event, and action. Triggers define the targets meeting some conditions that lead to events. Events trigger corresponding actions.

The trigger list defines what managed objects or targets are to be monitored and how and relates each trigger to an event. In this model, the trigger list provides three trigger conditions:

- Existence: When a specific yang instance appears, the trigger fires.
- Boolean: If the test result is true the trigger fires.
- Threshold: The event that may be triggered when a YANG instance at data-instance list is met the threshold.

Each trigger can be seen as a logical test that, if satisfied or evaluated to be true, cause the action to be carried out.

The target list defines managed objects that can be added to notifications based or be set to a new value on the trigger, the trigger test type, or the event that resulted in the actions.

The event list defines what happens when an event is triggered, i.e., trigger corresponding action, e.g.,sending a notification, setting a value to the managed object or both.

The action list consists of updates or innvcations on local managed object attributes and defines a set of actions which will be performed (e.g. notification, set, another event, etc) when corresponding event be triggered. The value to be set can use many variations on rule structure.

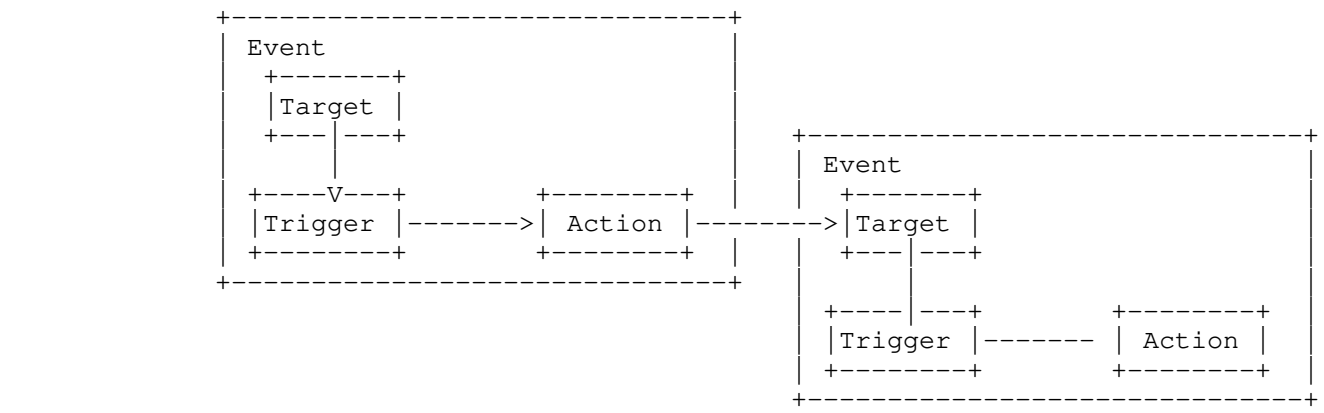
This document defines the YANG module "ietf-event", which has the following structure:

```
module: ietf-event
   +--rw events
      +--rw evt-smp-min?            uint32
      +--rw evt-smp-instance-max?   uint32
      +--rw event* [event-name type]
         +--rw event-name           string
         +--rw type                 identityref
         +--rw event-description?   string
         +--rw target*              target
         +--rw clear?               boolean
         +--rw related-event* [event-name type]
         |  +--rw event-name    string
         |  +--rw type          identityref
         +--rw trigger* [name]
         |  +--rw name                   string
         |  +--rw type?                  enumeration
```

```
|  +--rw trigger-description?   string
|  +--rw frequency
|  |  +--rw type?         identityref
|  |  +--rw periodic
|  |  |  +--rw interval    uint32
|  |  |  +--rw start?      yang:date-and-time
|  |  |  +--rw end?        yang:date-and-time
|  |  +--rw scheduling
|  |  |  +--rw month*           string
|  |  |  +--rw day-of-month*    uint8
|  |  |  +--rw day-of-week*     uint8
|  |  |  +--rw hour*            uint8
|  |  |  +--rw minute*          uint8
|  |  |  +--rw second*          uint8
|  |  |  +--rw start?           yang:date-and-time
|  |  |  +--rw end?             yang:date-and-time
|  |  +--rw immediate
|  |     +--rw immediate    empty
|  +--rw (test)?
|     +--:(existences)
|     |  +--rw existences
|     |     +--rw target*   -> /events/event/target
|     |     +--rw enable?   boolean
|     +--:(boolean)
|     |  +--rw boolean
|     |     +--rw comparison?   enumeration
|     |     +--rw value?        match-value
|     |     +--rw target*       -> /events/event/target
|     |     +--rw enable?       boolean
|     +--:(threshold)
|        +--rw threshold
|           +--rw rising-value?          match-value
|           +--rw rising-target*         -> /events/event/target
|           +--rw falling-value?         match-value
|           +--rw falling-target*        -> /events/event/target
|           +--rw delta-rising-value?    match-value
|           +--rw delta-rising-target*   -> /events/event/target
|           +--rw startup?               enumeration
+--rw action* [action-name]
   +--rw action-name           string
   +---n event-notification
   |  +---- event-name?   -> /events/event/event-name
   |  +---- type?         -> /events/event/type
   |  +---- target*       target
   +---x set
   |  +---w input
   |     +---w target*   target
   |     +---w value?    <anydata>
```

```
            +--rw trigger-event*        -> ../../event-name
```

The relation between Event, Trigger, Target and Action is described as follows:



One event may trigger another event, but if it does not trigger another event, the right part of above figure should be ignored.

7. EVENT YANG Module

<CODE BEGINS> file "ietf-event@2018-09-18.yang"

```
module ietf-event {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-event";
  prefix evt;

  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF xxx Working Group";
  contact
    "Zitao Wang: wangzitao@huawei.com
     Qin Wu: bill.wu@huawei.com";
  description
    "This module defines a model for the service topology.";

  revision 2018-09-18 {
    description
      "Initial revision.";
    reference "foo";
```

```
  }

  identity event-type {
    description
      "Base identity for event type";
  }

  identity frequency {
    description
      "Base identity for frequency";
  }

  identity periodic {
    base frequency;
    description
      "Identity for periodic trigger";
  }

  identity scheduling {
    base frequency;
    description
      "Identity for scheduling trigger";
  }

  identity immediate {
    base frequency;
    description
      "Identity for immediate trigger";
  }

  typedef match-value {
    type union {
      type yang:xpath1.0;
      type yang:object-identifier;
      type string;
    }
    description
      "This type is used to match resources of type 'target'.
       Since the type 'target' is a union of different types,
       the 'match-value' type is also a union of corresponding
       types.";
  }

  typedef target {
    type union {
      type instance-identifier;
      type yang:object-identifier;
      type yang:uuid;
```

```
      type string;
    }
    description
      "If the target is modelled in YANG, this type will
       be an instance-identifier.
       If the target is an SNMP object, the type will be an
       object-identifier.
       If the target is anything else, for example a distinguished
       name or a CIM path, this type will be a string.
       If the target is identified by a UUID use the uuid
       type.
       If the server supports several models, the presedence should
       be in the order as given in the union definition.";
  }

  grouping start-end-grouping {
    description
      "A grouping that provides start and end times for
       Event objects.";
    leaf start {
      type yang:date-and-time;
      description
        "The date and time when the Event object
         starts to create triggers.";
    }
    leaf end {
      type yang:date-and-time;
      description
        "The date and time when the Event object
         stops to create triggers.
         It is generally a good idea to always configure
         an end time and to refresh the end time as needed
         to ensure that agents that lose connectivity to
         their Controller do not continue executing Schedules
         forever.";
    }
  }

  container events {
    leaf evt-smp-min {
      type uint32;
      description
        "Sets the minimum value for data instance sampling";
    }
    leaf evt-smp-instance-max {
      type uint32;
      description
        "Sets the maximum value for data instance instance sampling.";
```

```
    }
    list event {
      key "event-name type";
      leaf event-name {
        type string;
        description
          "Event name";
      }
      leaf type {
        type identityref {
          base event-type;
        }
        description
          "Type of event";
      }
      leaf event-description {
        type string;
        description
          "Event description";
      }
      leaf-list target {
        type target;
        description
          "targeted objects";
      }
      leaf clear {
        type boolean;
        default "false";
        description
          "A flag indicate whether the event be closed";
      }
      list related-event {
        key "event-name type";
        leaf event-name {
          type string;
          description
            "Event name";
        }
        leaf type {
          type identityref {
            base event-type;
          }
          description
            "Type of event";
        }
        description
          "List for related events";
      }
```

```
      list trigger {
        key "name";
        leaf name {
          type string;
          description
            "Trigger name";
        }
        leaf type {
          type enumeration {
            enum "existence" {
              description
                "Indicates that the trigger type is 'existence'.
                 For 'existence', the specific test is as selected.
                 When an object appears, vanishes
                 or changes value, the trigger fires.";
            }
            enum "boolean" {
              description
                "Indicates that the trigger type is 'boolean'.
                 If the test result is true the trigger fires.  The
                 trigger will not fire again until the value has become false an
d
                 come back to true.";
            }
            enum "threshold" {
              description
                "Indicates that the trigger type is 'threshold'.
                 The event that may be triggered when a YANG instance
                 at data-instance list is found.  If the first sample after this
                 instance becomes active is greater than or equal to 'rising-val
ue'
                 and the 'startup' is equal to 'rising' then one threshold risin
g
                 event is triggered for that instance.";
            }
          }
          description
            "Trigger type";
        }
        leaf trigger-description {
          type string;
          description
            "Trigger description";
        }
        container frequency {
          leaf type {
            type identityref {
              base frequency;
            }
            description
              "Type of trigger frequency";
```

```
        }
        container periodic {
          when "derived-from-or-self(../type, 'periodic')";
          description
            "A periodic timing object triggers periodically
             according to a regular interval.";
          leaf interval {
            type uint32 {
              range "1..max";
            }
            units "seconds";
            mandatory true;
            description
              "The number of seconds between two triggers
               generated by this periodic timing object.";
          }
          uses start-end-grouping;
        }
        container scheduling {
          when "derived-from-or-self(../type, 'scheduling')";
          description
            "A scheduling timing object triggers.";
          leaf-list month {
            type string;
            description
              "A set of months at which this scheduling timing
               will trigger.";
          }
          leaf-list day-of-month {
            type uint8 {
              range "0..59";
            }
            description
              "A set of days of the month at which this
               scheduling timing will trigger.";
          }
          leaf-list day-of-week {
            type uint8 {
              range "0..59";
            }
            description
              "A set of weekdays at which this scheduling timing
               will trigger.";
          }
          leaf-list hour {
            type uint8 {
              range "0..59";
            }
```

```
            description
              "A set of hours at which the scheduling timing will
               trigger.";
          }
          leaf-list minute {
            type uint8 {
              range "0..59";
            }
            description
              "A set of minutes at which this scheduling timing
               will trigger.";
          }
          leaf-list second {
            type uint8 {
              range "0..59";
            }
            description
              "A set of seconds at which this calendar timing
               will trigger.";
          }
          uses start-end-grouping;
        }
        container immediate {
          when "derived-from-or-self(../type, 'immediate')";
          leaf immediate {
           type empty;
           mandatory true;
           description
            "This immediate Event object triggers immediately
             when it is configured.";
        }
        description
         "This immediate Event object triggers immediately
          when it is configured.";
       }
        description
          "Container for frequency";
      }
      choice test {
        description
          "Choice test";
        container existences {
          leaf-list target {
            type leafref {
              path "/events/event/target";
            }
            description
              "List for target objects";
```

```
            }
            leaf enable {
              type boolean;
              description
                "Startup";
            }
            description
              "Container for existence";
          }
          container boolean {
            leaf comparison {
              type enumeration {
                enum "unequal" {
                  description
                    "Indicates that the comparision type is 'unequal'.";
                }
                enum "equal" {
                  description
                    "Indicates that the comparision type is 'equal'.";
                }
                enum "leass" {
                  description
                    "Indicates that the comparision type is 'less'.";
                }
                enum "less-or-equal" {
                  description
                    "Indicates that the comparision type is 'less or equal'.";
                }
                enum "greater" {
                  description
                    "Indicates that the comparision type is 'greater'.";
                }
                enum "greater-or-equal" {
                  description
                    "Indicates that the comparision type is 'greater or equal'."
;
                }
              }
              description
                "Comparison type";
            }
            leaf value {
              type match-value;
              description
                "Compartion value";
            }
            leaf-list target {
              type leafref {
                path "/events/event/target";
```

```
            }
            description
              "List for target objects";
          }
          leaf enable {
            type boolean;
            description
              "Startup";
          }
          description
            "Container for boolean test";
        }
        container threshold {
          leaf rising-value {
            type match-value;
            description
              "Sets the rising threshold to the specified value, ]
               When the current sampled value is greater than or equal to
               this threshold, and the value at the last sampling interval
               was less than this threshold, the event is triggered. ";
          }
          leaf-list rising-target {
            type leafref {
              path "/events/event/target";
            }
            description
              "List for target objects";
          }
          leaf falling-value {
            type match-value;
            description
              "Sets the falling threshold to the specified value";
          }
          leaf-list falling-target {
            type leafref {
              path "/events/event/target";
            }
            description
              "List for target objects";
          }
          leaf delta-rising-value {
            type match-value;
            description
              "Sets the delta rising threshold to the specified value";
          }
          leaf-list delta-rising-target {
            type leafref {
              path "/events/event/target";
```

```
              }
              description
                "List for target objects";
            }
            leaf startup {
              type enumeration {
                enum "rising" {
                  description
                    "If the first sample after this
                     instance becomes active is greater than or equal to 'rising
-value'
                     and the 'startup' is equal to 'rising' then one threshold r
ising
                     event is triggered for that instance.";
                }
                enum "falling" {
                  description
                    "If the first sample
                     after this instance becomes active is less than or equal to
                     'falling-value' and the 'startup' is equal to 'falling' the
n one
                     threshold falling event is triggered for that instance.";
                }
                enum "rising-or-falling" {
                  description
                    "That event is
                       also triggered if the first sample after this entry becom
es
                       active is less than or equal or rising than to this thres
hold and
                       'startup' is equal to 'rising-or-falling'.";
                }
              }
              description
                "Startup";
            }
            description
              "Container for threshold";
          }
        }
        description
          "List for trigger";
      }
      list action {
        key "action-name";
        leaf action-name {
          type string;
          description
            "Action name";
        }
        notification event-notification {
          leaf event-name {
            type leafref {
```

```
            path "/events/event/event-name";
          }
          description
            "Report the event name";
        }
        leaf type {
          type leafref {
            path "/events/event/type";
          }
          description
            "Report the event type";
        }
        leaf-list target {
          type target;
          description
            "Report the target objects";
        }
        description
          "This notification is used to report that an operator
           acted upon an Event.";
      }
      action set {
        input {
          leaf-list target {
            type target;
            description
              "Report the target objects";
          }
          anydata value {
            description
              "Inline set content.";
          }
        }
      }
      leaf-list trigger-event {
        type leafref {
          path "../../event-name";
        }
        description
          "This action trigger another event";
      }
      description
        "List for Actions";
    }
    description
      "List for Events";
  }
  description
```

```
      "YANG data module for defining event triggers
       and actions for network management purposes";
  }
}
```

<CODE ENDS>

8. Security Considerations

The YANG modules defined in this document MAY be accessed via the RESTCONF protocol [RFC8040] or NETCONF protocol ([RFC6241]). The lowest RESTCONF or NETCONF layer requires that the transport-layer protocol provides both data integrity and confidentiality, see Section 2 in [RFC8040] and [RFC6241]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH)[RFC6242] . The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

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

- /events/event/event-name
- /events/event/target
- /events/action/set/target
- /events/event/trigger/name

9. IANA Considerations

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

```
---------------------------------------------------------------------
   URI: urn:ietf:params:xml:ns:yang:ietf-event
   Registrant Contact: The IESG.
   XML: N/A, the requested URI is an XML namespace.
---------------------------------------------------------------------
```

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

```
---------------------------------------------------------------------
   Name:         ietf-event
   Namespace:    urn:ietf:params:xml:ns:yang:ietf-event
   Prefix:       evt
   Reference:    RFC xxxx
---------------------------------------------------------------------
```

10. Normative References

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

[RFC2981] Kavasseri, R., Ed., "Event MIB", RFC 2981, DOI 10.17487/RFC2981, October 2000, <https://www.rfc-editor.org/info/rfc2981>.

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

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

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

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6370] Bocci, M., Swallow, G., and E. Gray, "MPLS Transport Profile (MPLS-TP) Identifiers", RFC 6370, DOI 10.17487/RFC6370, September 2011, <https://www.rfc-editor.org/info/rfc6370>.

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

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

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

Appendix A. Example of Event

For example, some service requires to monitoring the "in-errors" state of the interface, and if the value of "in-errors" exceeds the threshold, the event should reset the interface's enabled value to false:

```
<events>
 <event>
  <event-name>interface-state-exception</event-name>
  <type>interface-exception</type>
  <target>/if:interfaces/if:interface[if:name='eth1']</target>
  <target>/if:interfaces/if:interface[if:name='eth2']</target>
  <target>/if:interfaces/if:interface[if:name='eth3']</target>
  <trigger>
   <name>evaluate-in-errors</name>
   <trigger-description>evaluate the number of
    the packets that contained errors
    </trigger-description>
   <frequency>10m</frequency>
   <type>threshold</type>
   <test>
    <threshold>
     <startup>rising</startup>
     <rising-value>100</rising-value>
     <rising-target>/if:interfaces/if:interface[if:name='eth1']
       /if:statistic/if:in-errors</rising-target>
     <rising-target>/if:interfaces/if:interface[if:name='eth2']
       /if:statistic/if:in-errors</trising-arget>
    </threshold>
   </test>
  </trigger>
  <action>
   <name>interface-exception</name>
   <event-notification>
    <event-name>interface-state-exception</event-name>
    <type>interface-exception</type>
    <target>/if:interfaces/if:interface[if:name='eth1']</target>
   </event-notification>
   <set>
    <target>/if:interfaces/if:interface[if:name='eth1']</target>
    <interger-value>
     <interfaces>
      <interface>
       <name>eth1</name>
       <enable>false</enable>
      </interface>
     </interfaces>
    </interger-value>
   </set>
  <action>
 </event>
</events>
```

Authors' Addresses

Michael Wang
Huawei Technologies,Co.,Ltd
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: wangzitao@huawei.com

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

Email: bill.wu@huawei.com

Chongfeng Xie
China Telecom

Email: xiechf@ctbri.com.cn