

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 6, 2020

I. Bryskin
Futurewei
X. Liu
Volta Networks
A. Clemm
Huawei
H. Birkholz
Fraunhofer SIT
T. Zhou
Huawei
July 5, 2019

Generalized Network Control Automation YANG Model
draft-bryskin-netconf-automation-yang-03

Abstract

This document describes a YANG data model for the Generalized Network Control Automation (GNCA), aimed to define an abstract and uniform semantics for NETCONF/YANG scripts in the form of Event-Condition-Action (ECA) containers.

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 6, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	2
2. Purpose	3
3. GNCA concepts and constructs	4
3.1. Policy Variable (PV)	4
3.2. ECA Event	6
3.3. ECA Condition	7
3.4. ECA Action	9
3.5. ECA	11
4. Where ECA scripts are executed?	12
5. ECA script example	13
6. Complete Model Tree Structure	15
7. YANG Module	21
8. IANA Considerations	40
9. Security Considerations	40
10. Acknowledgements	40
11. References	41
11.1. Normative References	41
11.2. Informative References	41
Authors' Addresses	41

1. Introduction

NETCONF/YANG has proved to be very successful in facilitating interactive network control paradigm, in which a network control application (controller) requests the network server to perform certain (re-)configurations, then, retrieves network states of interest, then asks for more (re-)configurations and so forth. This said, there are multiple use cases that require stringing network configuration requests together with conditioning their order of execution based on certain events detected in the network, as well as current, historical or predicted network states, and pushing such request batches/scripts to the network server in order to control the network close loop automation processes. The Generalized Network Control Automation (GNCA) YANG model introduced by this document defines an abstract and uniform semantics of such NETCONF/YANG scripts in the form of Event-Condition-Action (ECA) containers.

2. Purpose

The purpose of the Generalized Network Control Automation (GNCA) YANG model is to enable an environment allowing for manipulation of close loop network automation via configuration of abstract Event-Condition-Action (ECA) scripts. The model defines the semantics of said scripts; however, how the scripts are actually implemented by the server is not governed by the model. The server may, for example, based on pushed ECA configuration, generate and execute server specific scripts. How this is done is outside of the scope of this document.

Although not all event response behaviors could be delegated to the network server, there are many circumstances where it is highly desirable. For example:

- a. Reaction to many network events is well understood and does not require additional information (such as broader or higher level network view or analytics input);
- b. It is often imperative for the network to start acting as quickly as the event is detected. In other words, there might simply be no time for the network-controller communication;
- c. A paradigm in which a controller micro-manages every network behavior does not scale. Simply put, there are always things that the network has to do autonomously (i.e. when the controller is not looking), which does not mean that the controller should have no say as to how said things need to be done;
- d. Numerous important use cases and applications can benefit from ability to define in an abstract and uniformly way a programmable logic in one place (e.g. on the controller) and execute it someplace else (e.g. on the server).

The main objective of the GNCA is to generalize the ECA network control style, so that it could be applied to arbitrary network events/conditions and manipulate network auto-control of large variety of network resources. Such a generalization would allow, for example, conveying to the network a coherent/ordered/prioritized behavior for recovering from a network failure or failures affecting simultaneously multiple services, instruct the network how to fight rapidly developing congestions, what to do when power outage is detected and so forth. In fact, it could be argued that without some sort of close loop network control automation hierarchical network control realistically could not be achieved.

The GNCA YANG model could be also thought of as an attempt to generalize the smart filter subscription machinery by stating that sending a notification is one of possibly multiple actions that network could perform reacting to the specified smart trigger. According to "Smart filters for Push Updates - Problem Statement" [I-D.clemm-netconf-push-smart-filters-ps]:

"They [smart filters] 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."

In a nutshell GNCA facilitates an environment in which the network automation logic could be defined in a form of ECA scripts by a client, pushed to the network before the events of interest happen, and executed by the network after the events are detected.

Finally, according to the smart filters problem statement, the following smart filters will be considered:

- "o Filters that involve freely programmable logic"

ECA scripts could serve as a vehicle to associate with a smart filter a complex freely programmable logic to detect the event of interest in the first place.

3. GNCA concepts and constructs

3.1. Policy Variable (PV)

Policy Variable (PV) is a structure to keep interim results/meta data during the execution of an ECA script. For example, a PV could be used as an output parameter of an RPC invoked by ECA1 to be used in a condition expression for ECA2.

With respect to ECAs the scope of a PV is either global or (ECA) local. Global PVs are permanent. They are kept in the top level PV container and are shared between all ECAs of the script. Local PVs are kept within the internal PV container of an ECA. Local PVs could be either dynamic - appear/disappear with start/stop of the ECA execution, or static - exist as long as the ECA is configured.

Each PV has the following attributes:

- o Globally or ECA scope unique name;

- o type - either pre-defined (e.g. Boolean, integer, uint64, etc.) or specified via XPath pointing to a data store node or a sub-tree of the required structure. For example, PV named "Link" could be associated with the "/TE_Topologies/TE_Links/TE_Link" XPath in accordance with the TE Topology model [I-D.ietf-teas-yang-te-topo] and hence be capable of accommodating the entire TE Link container;
- o value - data stored in the PV structured according to the PV type.

The following operations are allowed with/on a PV:

- o initialize (with a constant/enum/identity);
- o assign (with contents of another same type PV);
- o read (retrieve data store contents pointed by the specified same type XPath/sub-tree);
- o write (modify same type CONFIG=TRUE data store state with the PV's content/value);
- o insert (PV's content into a same type list);
- o iterate (copy into PV one by one same type list elements) (See examples of definition of and operations with PVs in Section 5).
- o function calls in a form of F(dst, src), where F is an identity of a function from extendable function library, dst and src are destination and source PVs respectively, the function's input parameters, with the result returned in dst.

Arbitrary expressions with PVs are for future study.

PVs could be used as input/output of an ECA invoked RPC. PVs could also be a source of information sent to the client in notification messages.

PVs could be used in condition expressions (see Section 5).

The model structure for the Policy Variable is shown below:

```

+--rw policy-variables
|   +--rw policy-variable* [name]
|       +--rw name          string
|       +--rw (type-choice)?
|           +--:(common)
|               |   +--rw type?      identityref
|           +--:(xpath)
|               +--rw xpath?    string
|   +--rw value?            <anydata>

```

3.2. ECA Event

ECA Event is any subscribable event notification either explicitly defined in a YANG module supported by the server or a smart filter conveyed to the server via smart filter subscription. Additionally, an event could be associated with a one-time or periodic timer.

Event notification contents become in the ECA context implicit local dynamic PVs with automatically generated names. Let's assume `Network_Failure_Is_Detected` event is defined that carries to a subscriber the detected failure type (`failureType`), ID (`failureID`) and the affected by the failure TE link state (`teLink`). In the context of the associated with the `Networ_Failure_Is_Detected` event ECA `failureType`, `failureID` and `teLink` are implicit local dynamic PVs that could be used in the embodied Condition and Action containers along with explicitly defined global and ECA local (static and/or dynamic) PVs.

One way to think of NETWONF/YANG Event Notification is as Remote Procedure Call-back, i.e. server->client directed RPC. When a subscribable NETWONF/YANG Event and associated with it ECA is pushed to the server within an ECA script, the server is expected to interpret this as follows: take the contents of the event notification and execute the logic defined by the associated Condition-Action chain (as I (client) would do on receipt of the notification) in order to decide how to react to the event. Recall that the whole purpose of ECA scripts is to reduce as much as possible the client-server communication.

A client may define an event of interest by making use of YANG PUSH smart filter/subscription. Specifically, the client may configure an ECA event according to the GNCA model specifying the event's name, as well as the name of corresponding PUSH subscription. In this case the server is expected to:

- o Register the event recording its name and using the referred PUSH subscription trigger as definition of the event firing trigger;

- o Auto-configure the event's ECA input in the form of local PVs using the PUSH subscription's filters;
- o At the moment of event firing intercept the notifications that would be normally sent to the PUSH subscription's client(s); copy the data store states pointed by the PUSH subscription's filters into the auto-configured ECA's local PVs and execute the ECA's condition-action chain.

All events (specified in at least one ECA pushed to the server) are required to be constantly monitored by the server. One way to think of this is that the server subscribes to its own publications with respect to all events that are associated with at least one ECA.

3.3. ECA Condition

ECA Condition is evaluated to TRUE or FALSE logical expression. There are two ways how an ECA Condition could be specified:

- o in a form of XPath expression;
- o as a hierarchy of comparisons and logical combinations of thereof.

The former option allows for specifying a condition of arbitrary complexity as a single string with an XPath expression, in which pertinent PVs and data store states are referred to by their respective positions in the YANG tree.

The latter option allows for configuring logical hierarchies. The bottom of said hierarchies are primitive comparisons (micro-conditions) specified in a form of:

`<arg1><relation><arg2>`

where arg1 and arg2 represent either constant/enum/identity, PV or pointed by XPath data store node or sub-tree,

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

Primitive comparisons could be combined hierarchically into macro-conditions via && and || logical operations.

Regardless of the choice of their specification, ECA Conditions are associated with ECA Events and evaluated only within event threads triggered by the event detection.

When an ECA Condition is evaluated to TRUE, the associated with it ECA Action is executed.

The model structure for the ECA Condition is shown below:

```

+--rw conditions
|   +--rw condition* [name]
|   |   +--rw name string
|   |   +--rw (expression-choice)?
|   |   |   +--:(logical-operation)
|   |   |   |   +--rw logical-operation-type? identityref
|   |   |   |   +--rw comparison-operation* [name]
|   |   |   |   |   +--rw name string
|   |   |   |   |   +--rw comparision-type? identityref
|   |   |   |   |   +--rw arg1
|   |   |   |   |   |   +--rw policy-argument
|   |   |   |   |   |   |   +--rw type? identityref
|   |   |   |   |   |   |   +--rw (argument-choice)?
|   |   |   |   |   |   |   |   +--:(policy-constant)
|   |   |   |   |   |   |   |   |   +--rw constant? string
|   |   |   |   |   |   |   |   +--:(policy-variable)
|   |   |   |   |   |   |   |   |   +--rw policy-variable? leafref
|   |   |   |   |   |   |   |   +--:(local-policy-variable)
|   |   |   |   |   |   |   |   |   +--rw local-policy-variable? leafref
|   |   |   |   |   |   |   |   +--:(xpath)
|   |   |   |   |   |   |   |   |   +--rw xpath? string
|   |   |   |   |   +--rw arg2
|   |   |   |   |   |   +--rw policy-argument
|   |   |   |   |   |   |   +--rw type? identityref
|   |   |   |   |   |   |   +--rw (argument-choice)?
|   |   |   |   |   |   |   |   +--:(policy-constant)
|   |   |   |   |   |   |   |   |   +--rw constant? string
|   |   |   |   |   |   |   |   +--:(policy-variable)
|   |   |   |   |   |   |   |   |   +--rw policy-variable? leafref
|   |   |   |   |   |   |   |   +--:(local-policy-variable)
|   |   |   |   |   |   |   |   |   +--rw local-policy-variable? leafref
|   |   |   |   |   |   |   |   +--:(xpath)
|   |   |   |   |   |   |   |   |   +--rw xpath? string
|   |   |   |   |   +--rw sub-condition* [name]
|   |   |   |   |   |   +--rw name -> /gnca/conditions/condition/name
|   |   |   |   +--:(xpath)
|   |   |   +--rw condition-xpath? string

```

The policy arguments arg1 and arg2 have the following structure:


```

+--rw policy-argument
  +--rw type? identityref
  +--rw (argument-choice)?
    +--:(policy-constant)
      | +--rw constant? string
    +--:(policy-variable)
      | +--rw policy-variable? leafref
    +--:(local-policy-variable)
      | +--rw local-policy-variable? leafref
    +--:(xpath)
      +--rw xpath? string

```

3.4. ECA Action

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

- o (-re)configuration - modifying a CONFIG=TRUE data store state
- o (re-)configuration scheduling - scheduling one time or periodic (re-)configuration in the future
- o sending one time notification;
- o adding/removing event notify subscription (essentially, the same action as performed when a client explicitly adds/removes a subscription)
- o executing an RPC defined by a YANG module supported by the server (the same action as performed when a client interactively calls the RPC);
- o performing operations and function calls on PVs (such as assign, read, insert, iterate, etc);
- o starting/stopping timers;
- o stopping current ECA;
- o invoking another ECA;
- o NO-ACTION action - meaningful only within ECA's Cleanup Condition-Action list to indicate that the ECA's Normal Condition-Action thread must be terminated as soon as one of the required Actions is rejected by the server (see more Section 3.4).

Two points are worth noting:

1. When a NETCONF/YANG RPC appears in an ECA Action body, the server is expected to interpret this as follows: execute the same logic, as when the client explicitly calls said RPC via NETCONF. For example, when TE_Tunnel_Path_Computation RPC is found in the currently executed Action, the server is expected to call its TE path computation engine and pass to it the specified parameters in the Action input.
2. When a "Send notification" action is configured as an ECA Action, the notification message to be sent to the client may contain not only elements of the data store (as, for example, YANG PUSH or smart filter notifications do), but also the contents of global and local PVs, which store results of arbitrary operations performed on the data store contents (possibly over arbitrary period of time) to determine, for example, history/evolution of data store changes, median values, ranges and rates of the changes, results of configured function calls and expressions, etc. - in short, any data the client may find interesting about the associated event with all the logic to compute said data delegated to the server. Importantly, ECA notifications are the only ECA actions that directly interact with and hence need to be unambiguously understood by the client. Furthermore, the same ECA may originate numerous single or repetitive semantically different notifications within the same or separate event firings. In order to facilitate for the client the correlation of events and ECA notifications received from the server, the GNCA model requires each notification to carry mandatory information, such as event and (event scope unique) notification names.

Multiple ECA Condition/Action pairs could be combined into a macro-action.

Multiple ECA (macro-)Actions could be triggered by a single ECA event.

Any given ECA Condition or Action may appear in more than one ECAs.

The model structure for the ECA Action is shown below:

```

+--rw actions
|   +--rw action* [name]
|   |   +--rw name string
|   |   +--rw action-element* [name]
|   |   |   +--rw name string
|   |   |   +--rw action-type? identityref
|   |   |   +--rw (action-operation)?
|   |   |   |   +--:(action)
|   |   |   |   |   +--rw action-name?
|   |   |   |   |   |   -> /gnca/actions/action/name
|   |   |   |   +--:(content-moving)
|   |   |   |   |   +--rw content-moving
|   |   |   |   |   |   +--rw content-moving-type? identityref
|   |   |   |   |   |   +--rw src
|   |   |   |   |   |   |   +--rw policy-argument
|   |   |   |   |   |   +--rw dst
|   |   |   |   |   |   |   +--rw policy-argument
|   |   |   |   +--:(function-call)
|   |   |   |   |   +--rw function-call
|   |   |   |   |   |   +--rw function-type? identityref
|   |   |   |   |   |   +--rw src
|   |   |   |   |   |   |   +--rw policy-argument
|   |   |   |   |   |   +--rw dst
|   |   |   |   |   |   |   +--rw policy-argument
|   |   |   |   +--:(rpc-operation)
|   |   |   |   |   +--rw rpc-operation
|   |   |   |   |   |   +--rw name? string
|   |   |   |   |   |   +--rw nc-action-xpath? string
|   |   |   |   |   |   +--rw policy-variable* [name]
|   |   |   |   |   |   |   +--rw name string
|   |   |   |   |   |   |   +--rw policy-argument
|   |   |   |   +--:(notify-operation)
|   |   |   |   |   +--rw notify-operation
|   |   |   |   |   |   +--rw name? string
|   |   |   |   |   |   +--rw policy-variable* [name]
|   |   |   |   |   |   |   +--rw name string
|   |   |   |   |   |   |   +--rw policy-argument
|   |   +--rw time-schedule
|   |   |   +--rw start? yang:date-and-time
|   |   |   +--rw repeat-interval? string

```

3.5. ECA

An ECA container includes:

- o Event name

- o List of local PVs. As mentioned, local PVs could be configured as dynamic (their instances appear/disappear with start/stop of the ECA execution) or static (their instances exist as long as the ECA is configured). The ECA input (the contents of the associated notification message, such as YANG PUSH or smart filter notification message) are the ECA's implicit local dynamic PVs with automatically generated names
- o Normal CONDITION-ACTION list: configured conditions each with associated actions to be executed if the condition is evaluated to TRUE
- o Cleanup CONDITION-ACTION list: configured conditions/actions to be evaluated/executed in case any Action from the Normal CONDITION-ACTION list was attempted and rejected by the server. In other words, this list specifies the ECA cleanup/unroll logic after rejection of an Action from the Normal CONDITION-ACTION list. An empty Cleanup CONDITION-ACTION list signifies that the ECA's normal Actions should be executed regardless of whether the previously attempted ECA Actions were rejected or not by the server. Cleanup CONDITION-ACTION list containing a single NO-ACTION Action signifies that the ECA thread is to be immediately terminated on rejection of any attempted Action (without executing any cleanup logic)

4. Where ECA scripts are executed?

It could be said that the main idea of the GNCA is decoupling the place where the network control logic is defined from the place where it is executed. In previous sections of this document it is assumed that the network control logic is defined by a client and pushed to and executed by the network (server). This is accomplished via ECA scripts, which are essentially bunches of regular NETCONF style operations (such as get, set, call rpc) and event notifications glued together via Policy Variables, PVs. It is worth noting that said ECA scripts could be easily moved around and executed by any entity supporting the GNCA YANG model (i.e. capable of interpreting the ECA scripts). One interesting implication of this is that the ECA scripts could be executed by neither client nor server, but a 3d party entity, for instance, with a special focus on the control of a particular network domain or/and special availability of/proximity to information/ resources that could contribute to the network control decision process. For example, the ECA scripts could be pushed to a Path Computation Element (PCE) adopted to support the GNCA YANG model. Specialized ECA scripts could be fanned out to multiple specialized controllers to take care of different aspects of a network domain control.

Another interesting idea is to combine the GNCA with hierarchical T-SDN architecture. In particular, the ECA scripts conveyed by a client to a network orchestrator could be pushed (modified or unmodified) hierarchically down to lower level controllers. After all, the goal of the hierarchical T-SDN is to create a paradigm in which the higher level of a controller in the hierarchy, the broader (topologically and/or functionally) its control on the network and the lesser its involvement in the network's micro-management in real time. On the other hand, it is desirable for a higher level controller to have a say as to how the subordinate controllers and, by extension, the network under control should deal with events and situations that are handled autonomously (i.e. without bothering the higher level controller in real time). The ECA scripts pushed down the T-SDN hierarchy may help to achieve this objective.

5. ECA script example

Consider a situation on a TE network when a network failure simultaneously affecting multiple TE tunnels. Normally, the TE network relies in this case on TE tunnels pre-configured protection/restoration capabilities and lets the TE tunnels to auto-recover themselves independently from each other. However, this default behavior may not be desired in some configurations/use cases because:

- a. Recovery procedures of a "greedy" TE tunnel may block the recovery of other TE tunnels;
- b. Shared mesh protection/restoration schemes are in place

In such cases the network has to perform the recovery of failure affected TE tunnels as a coordinated process. Furthermore, it is quite common that network resources available for the dynamic recovery procedures are limited, in which case it is desirable to convey to the network the policy/order in which the TE tunnels should be recovered. Different policies may be considered, to name a few:

1. Recover as many TE tunnels as possible;
2. Recover TE tunnels in accordance with their importance/priority;
3. Recover all unprotected TE tunnels before recovering broken connections/LSPs of protected TE tunnels (because the latter can tolerate the failure hopefully until it is repaired).

Let's describe an ECA script that could be pushed by the controller application instructing the network to perform multiple TE tunnel failure recovery according to policy (3) above.

Assumptions: it is assumed that in one or several YANG modules supported by the server the following is defined:

- o Subscribable "Network_Failure_Is_Detected" event carrying in the notification message the detected failure type (failureType), ID (failureID) and the affected by the failure TE link ID (linkID);
- o RPC "PathDependsOnLink" taking as an input a TE_Path and TE_Link_ID and returning in its output Boolean indicating whether or not the specified path goes through the link with the specified ID;
- o RPC "ReplaceTunnelsAwayFromLink" taking as an input a list of TE tunnel key leafrefs and ID of to be avoided link, performing the tunnel replacement away from the link and returning no output.

Explicit (global) PVs:

- o name: Yes type: Boolean
- o name: lsp xpath: /TE_Tunnels/lsp/lsp
- o name tunnel xpath: /TE_Tunnels/tunnels/te_tunnel
- o name: unprotected_tunnels xpath: /TE_Tunnels/tunnels/te_tunnel/dependent_tunnels
- o name protected_tunnels xpath: /TE_Tunnels/tunnels/te_tunnel/dependent_tunnels

Actions:

name: PopulateTunnelLists

body:

```
lsp iterate xpath:/TE_Tunnels/lsp
{
  rpc: PathDependsOnLink(<lsp>/rro, Yes);
  if(Yes == TRUE )
  {
    tunnel = <lsp>/parent_tunnel;
    if(<tunnel>/protectionType == UNPROTECTED)
      <tunne>/tunnelName insert unprotected_tunnels
    if(<tunnel>/protectionType != UNPROTECTED)
      <tunne>/tunnelName insert protected_tunnels
  }
}
```

name: RepairTunnels

Body:

```
rpc: ReplaceTunnelsAwayFromLink(unprotected_tunnels, linkID);
rpc: ReplaceTunnelsAwayFromLink(protected_tunnels, linkID);
```

ECA:

eventName: Network_Failure_Is_Detected;

eventParams: failureType, failureID, linkID

Condition: TRUE (always, every time)

Actions:

unprotected_tunnels = 0; protected_tunnels =0;

namedAction:PopulateTunnelLists;

namedAction:RepairTunnels

Note: RPC "PathDependsOnLink" is used in the example for simplicity. The RPC could be easily replaced by a scripted named action similar to PopulateTunnelLists .

6. Complete Model Tree Structure

The complete tree structure of the YANG model defined in this document is as follows:

```

module: ietf-gnca
+--rw gnca
  +--rw policy-variables
    +--rw policy-variable* [name]
      +--rw name string
      +--rw (type-choice)?
        +--:(common)
          +--rw type? identityref
        +--:(xpath)
          +--rw xpath? string
      +--rw value? <anydata>
  +--rw events
    +--rw event* [name]
      +--rw name string
      +--rw policy-variable?
        | -> /gnca/policy-variables/policy-variable/name
      +--rw local-policy-variable?
        | -> /gnca/ecas/eca/policy-variable/name
      +--rw (type-choice)?
        +--:(stream)
          +--rw stream
            +--rw name? string
            +--rw filter* string
            +--rw remote-publisher!
              +--rw address? inet:ip-address-no-zone
              +--rw port? inet:port-number
              +--rw transport? sn:transport
          +--:(timer)
            +--rw time-schedule!
              +--rw start? yang:date-and-time
              +--rw repeat-interval? string
  +--rw conditions
    +--rw condition* [name]
      +--rw name string
      +--rw (expression-choice)?
        +--:(logical-operation)
          +--rw logical-operation-type? identityref
          +--rw comparison-operation* [name]
            +--rw name string
            +--rw comparision-type? identityref
            +--rw arg1
              +--rw policy-argument
                +--rw type?
                  | identityref
                +--rw (argument-choice)?
                  +--:(policy-constant)
                    | +--rw constant? string
                  +--:(policy-variable)

```



```

leafref      | | | | | +-rw policy-variable?
              | | | | | +--:(local-policy-variable)
leafref      | | | | | +-rw local-policy-variable?
              | | | | | +--:(xpath)
              | | | | | +-rw xpath? string
              |--rw arg2
              |--rw policy-argument
              |   +-rw type?
              |   | identityref
              |--rw (argument-choice)?
              |   +--:(policy-constant)
              |   | +-rw constant? string
              |   +--:(policy-variable)
              |   | +-rw policy-variable?
leafref      | | | | | +--:(local-policy-variable)
              | | | | | +-rw local-policy-variable?
leafref      | | | | | +--:(xpath)
              | | | | | +-rw xpath? string
              |--rw sub-condition* [name]
              |   +-rw name -> /gnca/conditions/condition/name
              +--:(xpath)
              |   +-rw condition-xpath? string
+--rw actions
  +--rw action* [name]
    +--rw name string
    +--rw action-element* [name]
      +--rw name string
      +--rw action-type? identityref
      +--rw (action-operation)?
        +--:(action)
          +--rw action-name?
            -> /gnca/actions/action/name
        +--:(content-moving)
          +--rw content-moving
            +--rw content-moving-type? identityref
            +--rw src
              +--rw policy-argument
                +--rw type?
                  | identityref
                +--rw (argument-choice)?
                  +--:(policy-constant)
                  | +-rw constant? string
                  +--:(policy-variable)

```

				+++rw policy-variable?
leafref				+++:(local-policy-variable)
				+++rw local-policy-variable?
leafref				+++:(xpath)
				+++rw xpath?
				string
			+++rw dst	
			+++rw policy-argument	
			+++rw type?	
				identityref
			+++rw (argument-choice)?	
			+++:(policy-constant)	
				+++rw constant?
				string
			+++:(policy-variable)	
				+++rw policy-variable?
leafref				+++:(local-policy-variable)
				+++rw local-policy-variable?
leafref				+++:(xpath)
				+++rw xpath?
				string
			+++:(function-call)	
			+++rw function-call	
			+++rw function-type?	identityref
			+++rw src	
				+++rw policy-argument
				+++rw type?
				identityref
			+++rw (argument-choice)?	
			+++:(policy-constant)	
				+++rw constant?
				string
			+++:(policy-variable)	
				+++rw policy-variable?
leafref				+++:(local-policy-variable)
				+++rw local-policy-variable?
leafref				+++:(xpath)
				+++rw xpath?
				string
			+++rw dst	
			+++rw policy-argument	
			+++rw type?	

				identityref
				+--rw (argument-choice)?
				+--:(policy-constant)
				+--rw constant?
				string
				+--:(policy-variable)
				+--rw policy-variable?
leafref				
				+--:(local-policy-variable)
leafref				+--rw local-policy-variable?
				+--:(xpath)
				+--rw xpath?
				string
				+--:(rpc-operation)
				+--rw rpc-operation
				+--rw name? string
				+--rw nc-action-xpath? string
				+--rw policy-variable* [name]
				+--rw name string
				+--rw policy-argument
				+--rw type?
				identityref
				+--rw (argument-choice)?
				+--:(policy-constant)
				+--rw constant?
				string
				+--:(policy-variable)
				+--rw policy-variable?
leafref				
				+--:(local-policy-variable)
leafref				+--rw local-policy-variable?
				+--:(xpath)
				+--rw xpath?
				string
				+--:(notify-operation)
				+--rw notify-operation
				+--rw name? string
				+--rw policy-variable* [name]
				+--rw name string
				+--rw policy-argument
				+--rw type?
				identityref
				+--rw (argument-choice)?
				+--:(policy-constant)
				+--rw constant?
				string

```

|
|                                     +---:(policy-variable)
|                                     |  +---rw policy-variable?
leafref
|
|                                     +---:(local-policy-variable)
|                                     |  +---rw local-policy-variable?
leafref
|
|                                     +---:(xpath)
|                                     |  +---rw xpath?
|                                     |  string
|
|  +---rw time-schedule!
|      +---rw start?                yang:date-and-time
|      +---rw repeat-interval?     string
+---rw ecas
    +---rw eca* [name]
        +---rw name                string
        +---rw event-name          string
        +---rw policy-variable* [name]
            +---rw name            string
            +---rw (type-choice)?
            |   +---:(common)
            |   |   +---rw type?    identityref
            |   +---:(xpath)
            |       +---rw xpath?    string
            +---rw value?           <anydata>
            +---rw is-static?       boolean
        +---rw condition-action* [name]
            +---rw name            string
            +---rw condition?      -> /gnca/conditions/condition/name
            +---rw action?         -> /gnca/actions/action/name
        +---rw cleanup-condition-action* [name]
            +---rw name            string
            +---rw condition?      -> /gnca/conditions/condition/name
            +---rw action?         -> /gnca/actions/action/name
        +---x start
        +---x stop
        +---x pause
        +---x resume
        +---x next-action
        +---ro execution* [id]
            +---ro id                uint32
            +---ro oper-status?      oper-status
            +---ro start-time?
            |   yang:date-and-time
            +---ro stop-time?
            |   yang:date-and-time
            +---ro next-scheduled-time?
            |   yang:date-and-time
            +---ro last-condition-action?

```

```

|         |         -> ../../condition-action/name
+---ro last-condition?
|         |         -> ../../condition-action/condition
+---ro last-action?
|         |         -> ../../condition-action/action
+---ro last-cleanup-condition-action?
|         |         -> ../../cleanup-condition-action/name
+---rw eca-scripts
|   +---rw eca-script* [script-name]
|     +---rw script-name      string
|     +---rw eca* [eca-name]
|       +---rw eca-name      -> /gnca/ecas/eca/name
+---rw running-script?
|       -> /gnca/eca-scripts/eca-script/script-name

notifications:
+---n eca-execution
+---ro oper-status      oper-status
+---ro name              string
+---ro policy-variable* [name]
|   +---ro name          string
|   +---ro policy-argument
|     +---ro type?              identityref
|     +---ro (argument-choice)?
|       +---:(policy-constant)
|         |   +---ro constant?      string
|       +---:(policy-variable)
|         |   +---ro policy-variable?  leafref
|       +---:(local-policy-variable)
|         |   +---ro local-policy-variable?
|         |       -> /gnca/ecas/eca/policy-variable/name
|       +---:(xpath)
|         +---ro xpath?          string
+---ro value?                  <anydata>

```

7. YANG Module

```

<CODE BEGINS> file "ietf-gnca@2018-06-22.yang"
module ietf-gnca {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-gnca";

  prefix "gnca";

  import ietf-yang-types {

```

```
    prefix "yang";
}

import ietf-inet-types {
    prefix "inet";
}

import ietf-subscribed-notifications {
    prefix "sn";
}

organization
    "IETF Network Configuration (NETCONF) Working Group";

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

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

    Editor:     Xufeng Liu
                <mailto:xufeng.liu.ietf@gmail.com>

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

description
    "Event Condition Action (ECA) model.";

revision 2018-06-22 {
    description "Initial revision";
    reference "RFC XXXX";
}

/*
 * Typedefs
 */
identity argument-type {
    description
        "Possible values are:
        constant, variable, or datastore state.";
}

identity comparison-type {
    description
        "Possible values are:
        equal, not-equal, greater, greater-equal, less, less-equal.";
}
```

```
}

identity logical-operation-type {
    description
        "Possible values are:
        not, or, and.";
}

identity function-type {
    description
        "Possible values are:
        plus, minus, mult, divide, remain.";
}

identity content-moving-operation-type {
    description
        "Possible values are:
        copy, iterate, insert.";
}

identity action-type {
    description
        "Possible values are:
        action, content-move, function-call, rpc, notify.";
}

identity policy-variable-type {
    description
        "Possible values are:
        boolean, int32, int64, uint32, uint64, string, etc.";
}

/*
 * Typedefs
 */
typedef oper-status {
    type enumeration {
        enum completed {
            description "Completed with no error.";
        }
        enum running {
            description "Currently with no error.";
        }
        enum sleeping {
            description "Sleeping because of time schedule.";
        }
        enum paused {
            description "Paused by the operator.";
        }
    }
}
```

```
    }
    enum stoped {
        description "Stopped by the operator.";
    }
    enum failed {
        description "Failed with errors.";
    }
    enum error-handling {
        description
            "Asking the operator to handle an error.";
    }
}
description
    "The operational status of an ECA execution.";
}

/*
 * Groupings
 */
grouping policy-variable-attributes {
    description
        "Defining the policy variable attributes, including name, type
        and value. These attributes are used as part of the Policy
        Variable (PV) definition.";
    leaf name {
        type string;
        description
            "A string to uniquely identify a Policy Variable (PV), either
            globally for a global PV, or within the soope of ECA for a
            local PV.";
    }
    choice type-choice {
        description
            "The type of a policy variable may be either a common
            primitive type like boolean or a type from existing
            schema node referenced by an XPath string.";
        case common {
            leaf type {
                type identityref {
                    base policy-variable-type;
                }
                description
                    "A common policy variable type, defined as an
                    identity.";
            }
        }
        case xpath {
            leaf xpath {
```



```
        type string;
        description
            "A XPath string, referencing a schema node, whose
             type is used as the type of the policy variable.";
    }
}
}
anydata value {
    description
        "The value of the policy variable, in a format that is
         determined by the policy type.";
}
} // policy-variable-attributes

grouping policy-argument {
    description
        "Defining a policy argument, which can be used in a comparison
         or an action.";
    container policy-argument {
        description
            "Containing the attributes of a policy argument.";
        leaf type {
            type identityref {
                base argument-type;
            }
            description
                "Identifies the argument type.";
        }
        choice argument-choice {
            description
                "Argument formation options, depending on the policy
                 type.";
            case policy-constant {
                leaf constant {
                    type string;
                    description
                        "The constant value of the policy argument.";
                }
            }
            case policy-variable {
                leaf policy-variable {
                    type leafref {
                        path "/gnca/policy-variables/"
                          + "policy-variable/name";
                    }
                    description
                        "A reference to a global policy variable, which
                         is shared by all ECA scripts.";
                }
            }
        }
    }
}
```

```
    }
  }
  case local-policy-variable {
    leaf local-policy-variable {
      type leafref {
        path "/gnca/ecas/eca/policy-variable/name";
      }
      description
        "A reference to a local policy variable, which
         is kept within an ECA instance, and appears/
         disappears with start/stop of the ECA execution.";
    }
  }
  case xpath {
    leaf xpath {
      type string;
      description
        "An XPath string, referencing the data in the
         datastore.";
    }
  }
} // policy-argument

grouping action-element-attributes {
  description
    "Grouping of action element attributes.";
  leaf action-type {
    type identityref {
      base action-type;
    }
    description
      "Identifies the action type.";
  }
  choice action-operation {
    description
      "The operation choices that an ECA Action can take.";
    case action {
      leaf action-name {
        type leafref {
          path "/gnca/actions/action/name";
        }
        description
          "The operation is to execute a configured ECA Action.";
      }
    } // action
    case content-moving {
```

```
    container content-moving {
      description
        "The operation is to move contents between two policy
        arguments.";
      leaf content-moving-type {
        type identityref {
          base content-moving-operation-type;
        }
        description
          "The type of moving operation, which can be copy,
          iterate (copy a list of elements one by one), or
          insert.";
      }
      container src {
        description
          "The source policy argument.";
        uses policy-argument;
      }
      container dst {
        description
          "The destination policy argument.";
        uses policy-argument;
      }
    }
  } // content-moving
case function-call {
  container function-call {
    description
      "The operation is to call a function, which is of one of
      a few basic predefined types, such as plus, minus,
      multiply, devide, or remainder.";
    leaf function-type {
      type identityref {
        base function-type;
      }
      description
        "One of the predefined basic function types, such as
        plus, minus, multiply, devide, or remainder.";
    }
    container src {
      description
        "The source policy argument.";
      uses policy-argument;
    }
    container dst {
      description
        "The distination policy argument.";
      uses policy-argument;
    }
  }
}
```

```
    }
  }
} // function-call
case rpc-operation {
  container rpc-operation {
    description
      "The operation is to call an RPC, which is defined by
      a YANG module supported by the server.";
    leaf name {
      type string;
      description
        "The name of the YANG RPC or YANG action to be
        called.";
    }
    leaf nc-action-xpath {
      type string;
      description
        "The location where the YANG action is defined.
        This is used if and only if a YANG action is called.
        This leaf is not set when a YANG RPC is called.";
    }
    list policy-variable {
      key name;
      description
        "A list of policy arguments used as the input or output
        parameters passed to the RPC.";
      leaf name {
        type string;
        description
          "A string name used as the list key to form a list
          of policy arguments.";
      }
      uses policy-argument;
    }
  }
} // rpc-operation
case notify-operation {
  container notify-operation {
    description
      "The operation is to send a YANG notification.";
    leaf name {
      type string;
      description
        "Name of the subscribed YANG notification.";
    }
    list policy-variable {
      key name;
      description
```

```

        "A list of policy arguments carried in the notification
        message.";
    leaf name {
        type string;
        description
            "A string name used as the list key to form a list
            of policy arguments.";
    }
    uses policy-argument;
}
}
} // notify-operation
} // action-element-attributes

grouping time-schedule-container {
    description
        "Grouping to define a container of a time schedule.";
    container time-schedule {
        presence
            "Presence indicates that the timer is enabled.";
        description
            "Specifying the time schedule to execute an ECA Action, or
            trigger an event.";
        leaf start {
            type yang:date-and-time;
            description
                "The start time of the ECA Action, or the specified event.
                If not specified, the ECA Action is executed
                immediately when it is called, or the event is triggered
                immediately.";
        }
        leaf repeat-interval {
            type string {
                pattern
                    '(R\d*/)?P(\d+Y)?(\d+M)?(\d+W)?(\d+D)?T(\d+H)?'
                    + '(\d+M)?(\d+S)?';
            }
            description
                "The repeat interval to execute this ECA Action, or to
                trigger the event.
                The repeat interval is a string in ISO 8601 format,
                representing a delay duration or a repeated delay
                duration.
                If not specified, the ECA Action or the event trigger
                is executed without delay and without repetition.";
        }
    }
} // time-schedule

```

```
}

/*
 * Data nodes
 */
container gnca {
  description
    "Top level container for Generalized Network Control Automation
    (GNCA).";

  // policy-variables
  container policy-variables {
    description
      "Container of global Policy Variables (PVs).";
    list policy-variable {
      key name;
      description
        "A list of global Policy Variables (PVs), with a string
        name as the entry key.";
      uses policy-variable-attributes;
    }
  } // policy-variables

  container events {
    description
      "Container of ECA events.";
    list event {
      key name;
      description
        "A list of events used as the triggers of ECAs.";
      leaf name {
        type string;
        description
          "The name of the event.";
      }
      leaf policy-variable {
        type leafref {
          path "/gnca/policy-variables/"
            + "policy-variable/name";
        }
        description
          "Optional association to a global policy variable, which
          is shared by all ECA scripts.";
      }
      leaf local-policy-variable {
        type leafref {
          path "/gnca/ecas/eca/policy-variable/name";
        }
      }
    }
  }
}
```

```
description
  "Optional association to a local policy variable, which
   is kept within an ECA instance, and appears/
   disappears with start/stop of the ECA execution.";
}
choice type-choice {
  description
    "The type of an event, including subscribed stream.";
  case stream {
    container stream {
      description
        "The information of the subscribed stream.";
      leaf name {
        type string;
        description
          "The name of a stream subscribed and reported in
           the model
           ietf-subscribed-notifications.";
      }
      leaf-list filter {
        type string;
        description
          "A list of filters to be applied to the subscribed
           stream.";
      }
    }
    container remote-publisher {
      presence
        "The presence indicates the publisher is remote.
         Otherwise, the publisher is the local system.";
      description
        "If the subscribed stream is from a remote server,
         this container specifies the information of the
         remote server.";
      leaf address {
        type inet:ip-address-no-zone;
        description
          "IP address of the remote publisher.";
      }
      leaf port {
        type inet:port-number;
        description
          "The port number on the publisher for a
           subscription request RPC.";
      }
      leaf transport {
        type sn:transport;
        description
          "The transport used between this system and
```

```
        remote publisher.";
    }
}
}
case timer {
    uses time-schedule-container {
        description
            "Specifying the time schedule to trigger the event.
            If not specified, the event is not triggered.";
    }
}
}
}

container conditions {
    description
        "Container of ECA Conditions.";
    list condition {
        key name;
        description
            "A list of ECA Conditions.";
        leaf name {
            type string;
            description
                "A string name to uniquely identify an ECA Condition
                globally.";
        }
        choice expression-choice {
            description
                "The choices of expression format to specify a condition,
                which can be either a XPath string or a YANG logical
                operation structure.";
            case logical-operation {
                leaf logical-operation-type {
                    type identityref {
                        base logical-operation-type;
                    }
                    description
                        "The logical operation type used to combine the
                        results from the list comparison-operation and the
                        list sub-condition, defined below.";
                }
                list comparison-operation {
                    key name;
                    description
                        "A list of comparison operations, each of them defines
```


a comparison in the form of <arg1><relation><arg2>, where <arg1> and <arg2> are policy arguments, while <relation> is the comparison-type, which can be ==, !=, >, <, >=, <=;

```
leaf name {
  type string;
  description
    "A string name to uniquely identify a comparison
    operation.";
}
leaf comparison-type {
  type identityref {
    base comparison-type;
  }
  description
    "The comparison operation applied to the two
    arguments arg1 and arg2 defined below.";
}
container arg1 {
  description
    "The policy argument used as the first parameter of
    the comparison operation.
    A policy argument represents either a constant, PV
    or data store value pointed by XPath.";
  uses policy-argument;
}
container arg2 {
  description
    "The policy argument used as the second parameter
    of the comparison operation.
    A policy argument represents either a constant, PV
    or data store value pointed by XPath.";
  uses policy-argument;
}
list sub-condition {
  key name;
  description
    "A list of sub conditions applied by the
    logical-operation-type. This list of sub conditions
    provides the capability of hierarchically combining
    conditions.";
  leaf name {
    type leafref {
      path "/gnca/conditions/condition/name";
    }
    description
      "A reference to a defined condition, which is used
```

```
        as a sub-condition for the logical operation at
        this hierarchy level.";
    }
    } // sub-condition
} // logical-operation
case xpath {
    leaf condition-xpath {
        type string;
        description
            "A XPath string, representing a logical expression,
            which can contain comparisons of datastore values
            and logical operations in the XPath format.";
    }
    } // xpath
} // expression-choice
} // condition
} // conditions

container actions {
    description
        "Container of ECA Actions.";
    list action {
        key name;
        description
            "A list of ECA Actions.";
        leaf name {
            type string;
            description
                "A string name to uniquely identify an ECA Action
                globally.";
        }
    }

    list action-element {
        key name;
        description
            "A list of elements contained in an ECA Action. ";
        leaf name {
            type string;
            description
                "A string name to uniquely identify the action element
                within the scope of an ECA action.";
        }
        uses action-element-attributes;
    }

    uses time-schedule-container {
        description
            "Specifying the time schedule to execute this ECA
```

```
        Action.  
        If not specified, the ECA Action is executed immediately  
        when it is called.";   
    }  
}  
} // actions  
  
container ecas {  
    description  
        "Container of ECAs.";   
    list eca {  
        key name;  
        description  
            "A lis of ECAs";  
        leaf name {  
            type string;  
            description  
                "A string name to uniquely identify an ECA globally.";   
        }  
        leaf event-name {  
            type string;  
            mandatory true;  
            description  
                "The name of an event that triggers the execution of  
                this ECA.";   
        }  
    }  
  
    list policy-variable {  
        key name;  
        description  
            "A list of ECA local Policy Variables (PVs), with a  
            string name as the entry key.";   
        uses policy-variable-attributes;  
        leaf is-static {  
            type boolean;  
            description  
                "'true' if the PV is static; 'false' if the PV is  
                dynamic.  
                A dynamic PV appears/disappears with the start/stop  
                of the ECA execution; a static PV exists as long as  
                the ECA is configured.";   
        }  
    }  
}  
  
    list condition-action {  
        key name;  
        description  
            "A list of Condition-Actions, which are configured
```

```
        conditions each with associated actions to be executed
        if the condition is evaluated to TRUE";
    leaf name {
        type string;
        description
            "A string name uniquely identify a Condition-Action
            within this ECA.";
    }
    leaf condition {
        type leafref {
            path "/gnca/conditions/condition/name";
        }
        description
            "The reference to a configured condition.";
    }
    leaf action {
        type leafref {
            path "/gnca/actions/action/name";
        }
        description
            "The reference to a configured action.";
    }
} // condition-action

list cleanup-condition-action {
    key name;
    description
        "A list of Condition-Actions, which are configured
        conditions each with associated actions to be executed
        if the condition is evaluated to TRUE.
        This is the exception handler of this ECA, and is
        evaluated and executed in case any Action from the
        normal Condition-Action list was attempted and rejected
        by the server.";
    leaf name {
        type string;
        description
            "A string name uniquely identify a Condition-Action
            within this ECA.";
    }
    leaf condition {
        type leafref {
            path "/gnca/conditions/condition/name";
        }
        description
            "The reference to a configured condition.";
    }
    leaf action {
```

```
    type leafref {
      path "/gnca/actions/action/name";
    }
    description
      "The reference to a configured action.";
  }
} // cleanup-condition-action

action start {
  description
    "Start to execute this ECA.";
}
action stop {
  description
    "Stop the execution of this ECA.";
}
action pause {
  description
    "Pause the execution of this ECA.";
}
action resume {
  description
    "Resume the execution of this ECA.";
}
action next-action {
  description
    "Resume the execution of this ECA to complete the next
    action.";
}
list execution {
  key id;
  config false;
  description
    "A list of executions that this ECA has completed,
    are currently running, and will start in the scheduled
    future.";
  leaf id {
    type uint32;
    description
      "The ID to uniquely identify an execution of the ECA.";
  }
  leaf oper-status {
    type oper-status;
    description
      "The running status of the execution.";
  }
  leaf start-time {
    type yang:date-and-time;
  }
}
```

```
        description
            "The time when the ECA started.";
    }
    leaf stop-time {
        type yang:date-and-time;
        description
            "The time when the ECA completed or stopped.";
    }
    leaf next-scheduled-time {
        type yang:date-and-time;
        description
            "The next time when the ECA is scheduled to resume.";
    }
    leaf last-condition-action {
        type leafref {
            path "../..//condition-action/name";
        }
        description
            "The reference to a condition-action last executed
            or being executed.";
    }
    leaf last-condition {
        type leafref {
            path "../..//condition-action/condition";
        }
        description
            "The reference to a condition last executed or being
            executed.";
    }
    leaf last-action {
        type leafref {
            path "../..//condition-action/action";
        }
        description
            "The reference to aa action last executed or being
            executed.";
    }
    leaf last-cleanup-condition-action {
        type leafref {
            path "../..//cleanup-condition-action/name";
        }
        description
            "The reference to a cleanup-condition-action last
            executed or being executed.";
    }
}
} // ecas
```

```
container eca-scripts {
  description
    "Container of ECA Scripts.";
  list eca-script {
    key script-name;
    description
      "A list of ECA Script.";
    leaf script-name {
      type string;
      description
        "A string name to uniquely identify an ECA Script.";
    }
    list eca {
      key eca-name;
      description
        "A list of ECAs contained in this ECA Script.";
      leaf eca-name {
        type leafref {
          path "/gnca/ecas/eca/name";
        }
        description
          "The reference to a configured ECA.";
      }
    }
  }
} // eca-scripts

leaf running-script {
  type leafref {
    path "/gnca/eca-scripts/eca-script/script-name";
  }
  description
    "The reference to the ECA script that is currently running.";
}

/*
 * NOTIFICATIONS
 */

notification eca-execution {
  description
    "This notification is to send the result of an ECA execution
    that is specified to use notify-operation.";
  leaf oper-status {
    type oper-status;
    mandatory true;
  }
}
```

```
        description
            "The running status of the execution.";
    }
    leaf name {
        type string;
        mandatory true;
        description
            "Name of the subscribed YANG notification.";
    }
    list policy-variable {
        key name;
        description
            "A list of policy arguments carried in the notification
            message.";
        leaf name {
            type string;
            description
                "A string name used as the list key to form a list
                of policy arguments.";
        }
        uses policy-argument;
        anydata value {
            description
                "The value of the policy variable, in a format that is
                determined by the policy type.";
        }
    }
}
}
```

<CODE ENDS>

8. IANA Considerations

TBD.

9. Security Considerations

TBD.

10. Acknowledgements

The authors would like to thank Joel Halpern and Robert Wilton for their helpful comments and valuable contributions.

11. References

11.1. Normative References

- [I-D.clemm-netconf-push-smart-filters-ps]
Clemm, A., Voit, E., Liu, X., Bryskin, I., Zhou, T., Zheng, G., and H. Birkholz, "Smart filters for Push Updates - Problem Statement", draft-clemm-netconf-push-smart-filters-ps-00 (work in progress), October 2017.
- [I-D.ietf-teas-yang-te-topo]
Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Dios, "YANG Data Model for Traffic Engineering (TE) Topologies", draft-ietf-teas-yang-te-topo-22 (work in progress), June 2019.

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>>.
- [I-D.ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Event Notifications", draft-ietf-netconf-subscribed-notifications-26 (work in progress), May 2019.
- [I-D.ietf-netconf-yang-push]
Clemm, A. and E. Voit, "Subscription to YANG Datastores", draft-ietf-netconf-yang-push-25 (work in progress), May 2019.

Authors' Addresses

Igor Bryskin
Futurewei

Email: igor.bryskin@futurewei.com

Xufeng Liu
Volta Networks

Email: xufeng.liu.ietf@gmail.com

Alexander Clemm
Huawei

EMail: ludwig@clemm.org

Henk Birkholz
Fraunhofer SIT

EMail: henk.birkholz@sit.fraunhofer.de

Tianran Zhou
Huawei

EMail: zhoutianran@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 14, 2018

A. Clemm
Y. Qu
Huawei
J. Tantsura
Nuage Networks
A. Bierman
YumaWorks
June 12, 2018

Comparison of NMDA datastores
draft-clemm-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 December 14, 2018.

Copyright Notice

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

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

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

Table of Contents

1. Introduction	2
2. Key Words	3
3. Definitions and Acronyms	3
4. Data Model Overview	4
5. YANG Data Model	5
6. Example	8
7. Open Issues	10
8. Possible Future Extensions	10
9. IANA Considerations	11
9.1. Updates to the IETF XML Registry	11
9.2. Updates to the YANG Module Names Registry	11
10. Security Considerations	12
11. Acknowledgments	12
12. References	12
12.1. Normative References	12
12.2. Informative References	13
Authors' Addresses	13

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 possible 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:

- o source: The source identifies the datastore that will serve as reference for the comparison, for example <intended>.
- o target: The target identifies the datastore to compare against the source.
- o 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 does 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.)
- o 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:

- o 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@2018-06-12.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 2018-06-12 {
  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, and Kent Watsen 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-11 (work in progress), April 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

Internet-Draft

June 2018

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 30, 2021

R. Wilton, Ed.
D. Ball
T. Singh
Cisco Systems
S. Sivaraj
Juniper Networks
July 29, 2020

Common Interface Extension YANG Data Models
draft-ietf-netmod-intf-ext-yang-10

Abstract

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

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 30, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	3
1.1. Terminology	4
1.2. Tree Diagrams	4
2. Interface Extensions Module	4
2.1. Carrier Delay	5
2.2. Dampening	6
2.2.1. Suppress Threshold	7
2.2.2. Half-Life Period	7
2.2.3. Reuse Threshold	7
2.2.4. Maximum Suppress Time	7
2.3. Encapsulation	7
2.4. Loopback	8
2.5. Maximum frame size	8
2.6. Sub-interface	8
2.7. Forwarding Mode	9
3. Interfaces Ethernet-Like Module	9
4. Interface Extensions YANG Module	10
5. Interfaces Ethernet-Like YANG Module	21
6. Examples	25
6.1. Carrier delay configuration	25
6.2. Dampening configuration	26
6.3. MAC address configuration	27
7. Acknowledgements	29
8. ChangeLog	29
8.1. Version -10	29
8.2. Version -09	29
8.3. Version -08	29
8.4. Version -07	29
8.5. Version -06	29
8.6. Version -05	29
8.7. Version -04	29
8.8. Version -03	30
8.9. Version -02	30
9. IANA Considerations	30
9.1. YANG Module Registrations	30
10. Security Considerations	31
10.1. ietf-if-extensions.yang	31
10.2. ietf-if-ethernet-like.yang	32
11. References	32
11.1. Normative References	32

11.2. Informative References	33
Authors' Addresses	34

1. Introduction

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

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

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

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

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

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

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

1.1. Terminology

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

1.2. Tree Diagrams

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

2. Interface Extensions Module

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

The module provides:

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

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

```

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

```

2.1. Carrier Delay

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

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

this feature that is that any state transition will always be delayed by the specified time interval.

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

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

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

2.2. Dampening

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

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

2.2.1. Suppress Threshold

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

2.2.2. Half-Life Period

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

2.2.3. Reuse Threshold

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

2.2.4. Maximum Suppress Time

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

2.3. Encapsulation

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

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

2.4. Loopback

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

The following loopback modes are defined:

- o Internal loopback - All egress traffic on the interface is internally looped back within the interface to be received on the ingress path.
- o Line loopback - All ingress traffic received on the interface is internally looped back within the interface to the egress path.
- o Loopback Connector - The interface has a physical loopback connector attached that loops all egress traffic back into the interface's ingress path, with equivalent semantics to internal loopback.

2.5. Maximum frame size

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

2.6. Sub-interface

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

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

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

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

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

2.7. Forwarding Mode

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

The following forwarding modes are defined:

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

3. Interfaces Ethernet-Like Module

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

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

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

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

4. Interface Extensions YANG Module

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

```
<CODE BEGINS> file "ietf-if-extensions@2020-07-29.yang"
module ietf-if-extensions {
  yang-version 1.1;

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

  prefix if-ext;

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

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

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

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

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

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

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

description

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

Copyright (c) 2020 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

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

revision 2020-07-29 {

description

"Initial revision.";

reference

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

}

feature carrier-delay {

description

"This feature indicates that configurable interface carrier delay is supported, which is a feature is used to limit the propagation of very short interface link state flaps.";

reference "RFC XXXX, Section 2.1 Carrier Delay";

}

feature dampening {

description

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

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

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

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

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

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

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

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

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

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

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

/*
 * Augments the IETF interfaces model with leaves to configure
 * and monitor 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 capabilities may be rejected.";
}
leaf carrier-transitions {
  type yang:counter64;
  units transitions;
  config false;
  description
    "Defines the number of times the underlying carrier state
    has changed to, or from, state up.  This counter should be
    incremented even if the high layer interface state changes
    are being suppressed by a running carrier-delay timer.";
}
leaf timer-running {
  type enumeration {
    enum none {
      description
        "No carrier delay timer is running.";
    }
    enum up {
      description
        "Carrier-delay up timer is running.  The underlying
        carrier state is up, but interface state is not
        reported as up.";
    }
    enum down {
      description
        "Carrier-delay down timer is running.  Interface state
        is reported as up, but the underlying carrier state is
        actually down.";
    }
  }
  config false;
  description
    "Reports whether a carrier delay timer is actively running,
    in which case the interface state does not match the
    underlying carrier state.";
}

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

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

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

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

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



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

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

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

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

```
/*
 * Various types of interfaces support a configurable layer 2
 * encapsulation, any that are supported by YANG should be
 * listed here.
 *
 * Different encapsulations can hook into the common encaps-type
 * choice statement.
 */
container encapsulation {
  when
    "derived-from-or-self(..if:type,
                          'ianaift:ethernetCsmacd') or
     derived-from-or-self(..if:type,
                          'ianaift:ieee8023adLag') or
     derived-from-or-self(..if:type, 'ianaift:pos') or
     derived-from-or-self(..if:type,
                          'ianaift:atmSubInterface') or
     derived-from-or-self(..if:type, 'ianaift:l2vlan') or
     derived-from-or-self(..if:type, 'ethSubInterface')" {

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

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

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

```
    if-feature "loopback";
    type identityref {
        base loopback;
    }
    description "Enables traffic loopback.";
    reference "RFC XXXX, Section 2.4 Loopback";
}

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

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

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

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

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

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

```
*
* This should be extended to cover all interface types that are
* child interfaces of other interfaces.
*/
augment "/if:interfaces/if:interface" {
  when "derived-from(if:type, 'sub-interface') or
        derived-from-or-self(if:type, 'ianaift:l2vlan') or
        derived-from-or-self(if:type, 'ianaift:atmSubInterface') or
        derived-from-or-self(if:type, 'ianaift:frameRelay')" {
    description
      "Any ianaift:types that explicitly represent sub-interfaces
      or any types that derive from the sub-interface identity.";
  }
  if-feature "sub-interfaces";

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

    type if:interface-ref;

    mandatory true;
    description
      "This is the reference to the parent interface of this
      sub-interface.";
    reference "RFC XXXX, Section 2.6 Sub-interface";
  }
}

/*
* Add discard counter for unknown sub-interface encapsulation
*/
augment "/if:interfaces/if:interface/if:statistics" {
  when "derived-from-or-self(..if:type,
                              'ianaift:ethernetCsmacd') or
        derived-from-or-self(..if:type,
                              'ianaift:ieee8023adLag') or
        derived-from-or-self(..if:type, 'ianaift:ifPwType')" {
    description
      "Applies to interfaces that can demultiplex ingress frames to
      sub-interfaces.";
  }
  if-feature "sub-interfaces";

  description
    "Augment the interface model statistics with a sub-interface
    demux discard counter.";
```

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

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

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

5. Interfaces Ethernet-Like YANG Module

This YANG module augments the interface container defined in RFC 8343 [RFC8343] for Ethernet-like interfaces. This includes Ethernet interfaces, 802.3 LAG (802.1AX) interfaces, Switch Virtual interfaces, and Pseudo-Wire Head-End interfaces. It also contains references to [RFC6991], [RFC7224], and [IEEE802.3.2-2019].

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

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

  prefix ethlike;

  import ietf-interfaces {
```

```
    prefix if;
    reference
      "RFC 8343: A YANG Data Model For Interface Management";
  }

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

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

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

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

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

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

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

  Copyright (c) 2019 IETF Trust and the persons identified as
  authors of the code.  All rights reserved.

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

  This version of this YANG module is part of RFC XXXX
```

```
(https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
for full legal notices.";

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

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

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

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

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

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

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

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

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

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

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

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


<CODE ENDS>

6. Examples

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

6.1. Carrier delay configuration

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

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

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

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

6.2. Dampening configuration

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

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

6.3. MAC address configuration

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

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

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

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

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

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

7. Acknowledgements

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

8. ChangeLog

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

8.1. Version -10

- o Update modules from github and tree diagram.

8.2. Version -09

- o Fixed IANA section.

8.3. Version -08

- o Initial updates after WG LC comments.

8.4. Version -07

- o Minor editorial updates

8.5. Version -06

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

8.6. Version -05

- o Incorporate feedback from Andy Bierman

8.7. Version -04

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

8.8. Version -03

- o Fixed incorrect module name references, and updated tree output

8.9. Version -02

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

9. IANA Considerations

9.1. YANG Module Registrations

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

The ietf-if-extensions module:

Name: ietf-if-extensions

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

Prefix: if-ext

Reference: [RFCXXXX]

The ietf-if-ethernet-like module:

Name: ietf-if-ethernet-like

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

Prefix: ethlike

Reference: [RFCXXXX]

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

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

Registrant Contact: The IESG.

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

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

Registrant Contact: The IESG.

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

10. Security Considerations

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

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

10.1. ietf-if-extensions.yang

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

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

- o /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:

- o /if:interfaces/if:interface/carrier-delay/down
- o /if:interfaces/if:interface/carrier-delay/up
- o /if:interfaces/if:interface/dampening/half-life
- o /if:interfaces/if:interface/dampening/reuse

- o /if:interfaces/if:interface/dampening/suppress
- o /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:

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

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

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

10.2. ietf-if-ethernet-like.yang

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

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

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.

- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.

11.2. Informative References

- [I-D.ietf-netmod-sub-intf-vlan-model]
Wilton, R., Ball, D., tapsingh@cisco.com, t., and S. Sivaram, "Sub-interface VLAN YANG Data Models", draft-ietf-netmod-sub-intf-vlan-model-07 (work in progress), July 2020.
- [IEEE802.3.2-2019]
IEEE WG802.3 - Ethernet Working Group, "IEEE 802.3.2-2019", 2019.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<https://www.rfc-editor.org/info/rfc6536>>.

- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", RFC 7224, DOI 10.17487/RFC7224, May 2014, <<https://www.rfc-editor.org/info/rfc7224>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.

Authors' Addresses

Robert Wilton (editor)
Cisco Systems

Email: rwilton@cisco.com

David Ball
Cisco Systems

Email: daviball@cisco.com

Tapraj Singh
Cisco Systems

Email: tapsingh@cisco.com

Selvakumar Sivaraj
Juniper Networks

Email: ssivaraj@juniper.net

Network Working Group
Internet-Draft
Updates: 8407 (if approved)
Intended status: Standards Track
Expires: September 1, 2020

C. Hopps
LabN Consulting, L.L.C.
L. Berger
LabN Consulting, LLC.
D. Bogdanovic
Volta Networks
February 29, 2020

YANG Module Tags
draft-ietf-netmod-module-tags-10

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 registered and assigned during module definition; assigned by implementations; or dynamically defined and set by users. This document also provides guidance to future model writers; as such, this document updates RFC8407.

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 September 1, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	2
1.1. Some possible use cases for YANG module tags	3
1.2. Conventions Used in This Document	4
2. Tag Values	4
2.1. IETF Tags	4
2.2. Vendor Tags	4
2.3. User Tags	5
2.4. Reserved Tags	5
3. Tag Management	5
3.1. Module Definition Tagging	5
3.2. Implementation Tagging	5
3.3. User Tagging	5
4. Tags Module Structure	6
4.1. Tags Module Tree	6
4.2. YANG Module	6
5. Other Classifications	9
6. Guidelines to Model Writers	9
6.1. Define Standard Tags	9
7. IANA Considerations	9
7.1. YANG Module Tag Prefixes Registry	9
7.2. IETF YANG Module Tags Registry	10
7.3. Updates to the IETF XML Registry	12
7.4. Updates to the YANG Module Names Registry	12
8. Security Considerations	13
9. References	13
9.1. Normative References	13
9.2. Informative References	14
Appendix A. Examples	15
Appendix B. Acknowledgements	15
Appendix C. Non-NMDA State Module.	15
Authors' Addresses	18

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"). One benefit of using tags for organization over a rigid structure is that it is more flexible and can more easily adapt over time as technologies evolve. Tags can be usefully registered, but they can also serve as a non-registered mechanism

available for users to define themselves. This document provides a mechanism to define tags and associate them with YANG modules in a flexible manner. In particular, tags may be registered as well as assigned during module definition; assigned by implementations; or dynamically defined and set by users.

This document defines a YANG module [RFC7950] 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 6 provides guidelines for authors of YANG data models.

This document updates [RFC8407].

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

1.1. Some possible use cases for YANG module tags

During this documents's development there were requests for example uses of module tags. The following are a few example use cases for tags. This list is certainly not exhaustive.

One example use of tags would be to help filter different discrete categories of YANG modules supported by a device. For example, if modules are suitably tagged, then an XPath query can be used to list all of the vendor modules supported by a device.

Tags can also be used to help coordination when multiple semi-independent clients are interacting with the same devices. For example, one management client could mark that some modules should not be used because they have not been verified to behave correctly, so that other management clients avoid querying the data associated with those modules.

Tag classification is useful for users searching module repositories (e.g., YANG catalog). A query restricted to the 'ietf:routing' module tag could be used to return only the IETF YANG modules associated with routing. Without tags, a user would need to know the name of all the IETF routing protocol YANG modules.

Future management protocol extensions could allow for filtering queries of configuration or operational state on a server based on tags. For example, return all operational state related to system-management.

1.2. 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.

2. Tag Values

All tags SHOULD begin with a prefix indicating who owns their definition. An IANA registry (Section 7.1) is used to support registering tag prefixes. Currently 3 prefixes are defined. No further structure is imposed by this document on the value following the registered prefix, and the value can contain any YANG type 'string' characters except carriage-returns, newlines and tabs.

Again, except for the conflict-avoiding prefix, this document is not specifying any structure on (i.e., restricting) the tag values on purpose. The intent is to avoid arbitrarily restricting the values that designers, implementers and users can use. As a result of this choice, designers, implementers, and users are free to add or not add any structure they may require to their own tag values.

2.1. IETF Tags

An IETF tag is a tag that has the prefix "ietf:". All IETF tags are registered with IANA in a registry defined later in this document (Section 7.2).

2.2. 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 registered; however, it is RECOMMENDED that the vendor include extra identification in the tag to avoid collisions such as using the enterprise or organization name following the "vendor:" prefix (e.g., vendor:example.com:vendor-defined-classifier).

2.3. User Tags

A user tag is any tag that has the prefix "user:". These tags are defined by the user/administrator and are not meant to be registered. Users are not required to use the "user:" prefix; however, doing so is RECOMMENDED as it helps avoid collisions.

2.4. Reserved Tags

Any tag not starting with the prefix "ietf:", "vendor:" or "user:" is reserved for future use. These tag values are not invalid, but simply reserved in the context of specifications (e.g., RFCs).

3. 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.

3.1. Module Definition Tagging

A module definition MAY 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 is defined in an IETF standards track document, the tags MUST be IETF Tags (2.1). Thus, new modules can drive the addition of new IETF tags to the IANA registry defined in Section 7.2, and the IANA registry can serve as a check against duplication.

3.2. Implementation Tagging

An implementation MAY include additional tags associated with a module. These tags SHOULD be IETF Tags (i.e., registered) or vendor specific tags.

3.3. User Tagging

Tags of any kind, with or without a prefix, can be assigned and removed by the user using normal configuration mechanisms. In order to remove a tag from the operational datastore the user adds a matching "masked-tag" entry for a given module.

4. Tags Module Structure

4.1. 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
```

Figure 1: YANG Module Tags Tree Diagram

4.2. YANG Module

```
<CODE BEGINS> file "ietf-module-tags@2020-02-29.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
    "WG Web: <https://tools.ietf.org/wg/netmod/>
    WG List: <mailto:netmod@ietf.org>

    Author: Christian Hopps
            <mailto:chopps@chopps.org>

    Author: Lou Berger
            <mailto:lberger@labn.net>

    Author: Dean Bogdanovic
            <ivandean@gmail.com>";

  // 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) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

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

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

```
revision 2020-02-29 {
  description
    "Initial revision.";
  reference "RFC XXXX: YANG Module Tags";
}

typedef tag {
  type string {
    length "1..max";
    pattern '[\S ]+';
  }
  description
    "A tag is a type 'string' value that does not include carriage
    return, newline or tab characters. It SHOULD begin with a
    registered prefix; however, tags without a registered prefix
    SHOULD NOT be treated as invalid.";
}

extension 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'
        [RFC8342].";
    }

    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 Prefixes' registry for reserved prefixes and the IANA
                    'IETF YANG Module Tags' registry for IETF tags.

                    The 'operational' state [RFC8342] view of this list is
                    constructed using the following steps:

                    1) System tags (i.e., tags of 'system' origin) are added.
                    2) User configured tags (i.e., tags of 'intended' origin)
                       are added.
                    3) Any tag that is equal to a masked-tag is removed.";
            }
            leaf-list masked-tag {
                type tag;
                description
                    "The list of tags that should not be associated with this
                    module. The user can remove (mask) tags from the
                    operational state datastore [RFC8342] by adding them to
                    this list. It is not an error to add tags to this list
                    that are not associated with the module, but they have no
                    operational effect.";
            }
        }
    }
}
}
}
<CODE ENDS>
```

Figure 2: Module Tags Module

5. Other Classifications

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

6. Guidelines to Model Writers

This section updates [RFC8407].

6.1. Define Standard Tags

A module MAY 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 IETF standardized modules new tags MUST be assigned in the IANA registry defined below, see Section 7.2.

7. IANA Considerations

7.1. YANG Module Tag Prefixes Registry

IANA is asked to create a new registry "YANG Module Tag Prefixes" grouped under a new "Protocol" category named "YANG Module Tags".

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

Prefix entries in this registry should be short strings consisting of lowercase ASCII alpha-numeric characters and a final ":" character.

The allocation policy for this registry is Specification Required [RFC8126]. The Reference and Assignee values should be sufficient to identify and contact the organization that has been allocated the prefix.

The initial values for this registry are as follows.

Prefix	Description	Reference	Assignee
ietf:	IETF Tags allocated in the IANA IETF YANG Module Tags registry.	[This document]	IETF
vendor:	Non-registered tags allocated by the module implementer.	[This document]	IETF
user:	Non-registered tags allocated by and for the user.	[This document]	IETF

Other standards organizations (SDOs) wishing to allocate their own set of tags should allocate a prefix from this registry.

7.2. IETF YANG Module Tags Registry

IANA is asked to create a new registry "IETF YANG Module Tags" grouped under a new "Protocol" category "IETF YANG Module Tags". This registry should be included below "YANG Module Tag Prefixes" when listed on the same page.

This registry allocates tags that have the registered prefix "ietf:". New values should be well considered and not achievable through a combination of already existing IETF tags. IANA assigned tags must conform to Net-Unicode as defined in [RFC5198] and they shall not need normalization.

The allocation policy for this registry is IETF Review [RFC8126].

The initial values for this registry are as follows.

Tag	Description	Reference
ietf:network-element-class	[RFC8199] network element.	[RFC8199]
ietf:network-service-class	[RFC8199] network service.	[RFC8199]

ietf:sdo-defined-class	Module is defined by a standards organization.	[RFC8199]
ietf:vendor-defined-class	Module is defined by a vendor.	[RFC8199]
ietf:user-defined-class	Module is defined by the user.	[RFC8199]
ietf:hardware	Relates to hardware (e.g., inventory).	[This document]
ietf:software	Relates to software (e.g., installed OS).	[This document]
ietf:protocol	Represents a protocol (often combined with another tag to refine).	[This document]
ietf:qos	Relates to quality of service.	[This document]
ietf:network-service-app	Relates to a network service application (e.g., an NTP server, DNS server, DHCP server, etc).	[This document]
ietf:system-management	Relates to system management (e.g., a system management protocol such as syslog, TACAC+, SNMP, netconf, ...).	[This document]
ietf:oam	Relates to Operations, Administration, and Maintenance (e.g., BFD).	[This document]
ietf:routing	Relates to routing.	[This document]
ietf:security	Related to security.	[This document]
ietf:signaling	Relates to control plane signaling.	[This document]

ietf:link-management	Relates to link management.	[This document]
----------------------	-----------------------------	-----------------

7.3. Updates to the IETF XML Registry

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in [RFC3688], the following registrations have been made:

URI:

urn:ietf:params:xml:ns:yang:ietf-module-tags

Registrant Contact:

The IESG.

XML:

N/A; the requested URI is an XML namespace.

URI:

urn:ietf:params:xml:ns:yang:ietf-module-tags-state

Registrant Contact:

The IESG.

XML:

N/A; the requested URI is an XML namespace.

7.4. Updates to the YANG Module Names Registry

This document registers two YANG modules in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registration have been made:

name:

ietf-module-tags

namespace:

urn:ietf:params:xml:ns:yang:ietf-module-tags

prefix:

tags

reference:

RFC XXXX (RFC Ed.: replace XXX with actual RFC number and remove this note.)

```
name:
  ietf-module-tags-state

namespace:
  urn:ietf:params:xml:ns:yang:ietf-module-tags-state

prefix:
  tags

reference:
  RFC XXXX (RFC Ed.: replace XXX with actual RFC number and remove
  this note.)
```

8. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242].

This document adds the ability to associate tag meta-data with YANG modules. This document does not define any actions based on these associations, and none are yet defined, and therefore it does not by itself introduce any new security considerations directly.

Users of the tag-meta data may define various actions to be taken based on the tag meta-data. These actions and their definitions are outside the scope of this document. Users will need to consider the security implications of any actions they choose to define, including the potential for a tag to get 'masked' by another user.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [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>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <<https://www.rfc-editor.org/info/rfc8407>>.

9.2. Informative References

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, DOI 10.17487/RFC5198, March 2008, <<https://www.rfc-editor.org/info/rfc5198>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.

Appendix A. Examples

The following is a fictional NETCONF example result from a query of the module tags list. For the sake of brevity only a few module results are imagined.

```
<ns0:data xmlns:ns0="urn:ietf:params:xml:ns:netconf:base:1.0">
  <t:module-tags xmlns:t="urn:ietf:params:xml:ns:yang:ietf-module-tags">
    <t:module>
      <t:name>ietf-bfd</t:name>
      <t:tag>ietf:network-element-class</t:tag>
      <t:tag>ietf:oam</t:tag>
      <t:tag>ietf:protocol</t:tag>
      <t:tag>ietf:sdo-defined-class</t:tag>
    </t:module>
    <t:module>
      <t:name>ietf-isis</t:name>
      <t:tag>ietf:network-element-class</t:tag>
      <t:tag>ietf:protocol</t:tag>
      <t:tag>ietf:sdo-defined-class</t:tag>
      <t:tag>ietf:routing</t:tag>
    </t:module>
    <t:module>
      <t:name>ietf-ssh-server</t:name>
      <t:tag>ietf:network-element-class</t:tag>
      <t:tag>ietf:protocol</t:tag>
      <t:tag>ietf:sdo-defined-class</t:tag>
      <t:tag>ietf:system-management</t:tag>
    </t:module>
  </t:module-tags>
</ns0:data>
```

Figure 3: Example NETCONF Query Output

Appendix B. Acknowledgements

Special thanks to Robert Wilton for his help improving the introduction and providing the example use cases, as well as generating the non-NMDA module.

Appendix C. Non-NMDA State Module.

As per [RFC8407] the following is a non-NMDA module to support viewing the operational state for non-NMDA compliant servers.

```
<CODE BEGINS> file "ietf-module-tags-state@2020-02-29.yang"
module ietf-module-tags-state {
  yang-version 1.1;
```

```
namespace "urn:ietf:params:xml:ns:yang:ietf-module-tags-state";
prefix tags-s;
```

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

```
organization
```

```
"IETF NetMod Working Group (NetMod)";
```

```
contact
```

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

```
Author: Christian Hopps
       <mailto:chopps@chopps.org>
```

```
Author: Lou Berger
       <mailto:lberger@labn.net>
```

```
Author: Dean Bogdanovic
       <ivandean@gmail.com>;
```

```
// 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.
```

```
This is a temporary non-NMDA module that is for use by
implementations that don't yet support NMDA.
```

```
Copyright (c) 2019 IETF Trust and the persons identified as
authors of the code. All rights reserved.
```

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

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

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

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

```
revision 2020-02-29 {  
  description  
    "Initial revision.";  
  reference  
    "RFC XXXX: YANG Module Tags";  
}
```

```
container module-tags-state {  
  config false;  
  status deprecated;  
  description  
    "Contains the list of modules and their associated tags";  
  list module {  
    key "name";  
    status deprecated;  
    description  
      "A list of modules and their associated tags";  
    leaf name {  
      type yang:yang-identifier;  
      mandatory true;  
      status deprecated;  
      description  
        "The YANG module name.";  
    }  
    leaf-list tag {  
      type tags:tag;  
      status deprecated;  
      description  
        "Tags associated with the module. See the IANA 'YANG Module  
        Tag Prefixes' registry for reserved prefixes and the IANA  
        'IETF YANG Module Tags' registry for IETF tags.  
  
        The contents of this list is constructed using the  
        following steps:  
  
        1) System tags (i.e., tags of added by the system) are added.  
        2) User configured tags (i.e., tags added by configuration)  
        are added.  
        3) Any tag that is equal to a masked-tag present in the
```

```
        corresponding ietf-module-tags:module-tags:module-tag leaf
        list for this module is removed.";
    }
}
}
}
<CODE ENDS>
```

Figure 4: Non-NMDA Module Tags State Module

Authors' Addresses

Christian Hopps
LabN Consulting, L.L.C.

Email: chopps@chopps.org

Lou Berger
LabN Consulting, LLC.

Email: lberger@labn.net

Dean Bogdanovic
Volta Networks

Email: ivandean@gmail.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 14, 2021

R. Wilton, Ed.
D. Ball
T. Singh
Cisco Systems
S. Sivaraj
Juniper Networks
July 13, 2020

Sub-interface VLAN YANG Data Models
draft-ietf-netmod-sub-intf-vlan-model-07

Abstract

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

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

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 14, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
1.1. Terminology	3
1.2. Tree Diagrams	4
2. Objectives	4
2.1. Interoperability with IEEE 802.1Q compliant bridges	4
3. Interface VLAN Encapsulation Model	4
4. Interface Flexible Encapsulation Model	5
5. VLAN Encapsulation YANG Module	7
6. Flexible Encapsulation YANG Module	11
7. Examples	21
7.1. Layer 3 sub-interfaces with IPv6	22
7.2. Layer 2 sub-interfaces with L2VPN	23
8. Acknowledgements	26
9. ChangeLog	26
9.1. WG version -07 and -06	26
9.2. WG version -05	26
9.3. WG version -04	26
9.4. WG version -03	27
9.5. WG version -02	27
9.6. WG version -01	27
9.7. Version -04	27
9.8. Version -03	27
10. IANA Considerations	27
10.1. YANG Module Registrations	27
11. Security Considerations	28
11.1. ietf-if-vlan-encapsulation.yang	29
11.2. ietf-if-flexible-encapsulation.yang	29
12. References	31
12.1. Normative References	31
12.2. Informative References	32
Appendix A. Comparison with the IEEE 802.1Q Configuration Model	33

A.1. Sub-interface based configuration model overview	33
A.2. IEEE 802.1Q Bridge Configuration Model Overview	34
A.3. Possible Overlap Between the Two Models	34
Authors' Addresses	35

1. Introduction

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

The purpose of these models is to allow IETF defined forwarding protocols, for example, IPv6 [RFC2460], Ethernet Pseudo Wires [RFC4448] and VPLS [RFC4761] [RFC4762], when configured via appropriate YANG data models [RFC8344] [I-D.ietf-bess-l2vpn-yang], to interoperate with VLAN tagged traffic received from an IEEE 802.1Q compliant bridge.

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

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

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

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

ietf-if-flexible-encapsulation.yang - Defines the model for flexible classification of Ethernet/VLAN traffic, normally to L2 frame forwarding services

1.1. Terminology

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

The term 'sub-interface' is defined in section 2.6 of Interface Extensions YANG [I-D.ietf-netmod-intf-ext-yang].

1.2. Tree Diagrams

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

2. Objectives

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

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

2.1. Interoperability with IEEE 802.1Q compliant bridges

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

3. Interface VLAN Encapsulation Model

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

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

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

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

4. Interface Flexible Encapsulation Model

The Interface Flexible Encapsulation model is designed to allow for the flexible provisioning of layer 2 services. It provides the capability to classify and demultiplex Ethernet/VLAN frames received on an Ethernet trunk interface to sub-interfaces based on the fields available in the layer 2 headers. Once classified to sub-interfaces, it provides the capability to selectively modify fields within the layer 2 frame header before the frame is handed off to the appropriate forwarding code for further handling. The forwarding instance, e.g., L2VPN, VPLS, etc., is configured using a separate YANG configuration model defined elsewhere, e.g., [I-D.ietf-bess-l2vpn-yang].

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

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

The model also allows a flexible encapsulation and rewrite to be configured directly on an Ethernet or LAG interface without

configuring separate child sub-interfaces. Ingress frames that do not match the encapsulation are dropped. Egress frames MUST conform to the encapsulation.

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

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

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

```

module: ietf-if-flexible-encapsulation
  augment /if:interfaces/if:interface/if-ext:encapsulation
    /if-ext:encaps-type:
      +--:(flexible)
        +--rw flexible
          +--rw match
            +--rw (match-type)
              +--:(default)
                | +--rw default?          empty
              +--:(untagged)
                | +--rw untagged?          empty
              +--:(dot1q-priority-tagged)
                | +--rw dot1q-priority-tagged
                |   +--rw tag-type      dot1q-types:dot1q-tag-type
              +--:(dot1q-vlan-tagged)
                +--rw dot1q-vlan-tagged
                  +--rw outer-tag
                    +--rw tag-type      dot1q-tag-type
                    +--rw vlan-id        union
                  +--rw second-tag!
                    +--rw tag-type      dot1q-tag-type
                    +--rw vlan-id        union
                  +--rw match-exact-tags? empty
          +--rw rewrite {flexible-rewrites}?
            +--rw (direction)?
              +--:(symmetrical)
                +--rw symmetrical
                  +--rw dot1q-tag-rewrite {dot1q-tag-rewrites}?
                    +--rw pop-tags?      uint8
                    +--rw push-tags!

```

```

    +--rw outer-tag
    |   +--rw tag-type      dot1q-tag-type
    |   +--rw vlan-id      vlanid
    +--rw second-tag!
    |   +--rw tag-type      dot1q-tag-type
    |   +--rw vlan-id      vlanid
+--:(asymmetrical) {asymmetric-rewrites}?
+--rw ingress
|   +--rw dot1q-tag-rewrite {dot1q-tag-rewrites}?
|   +--rw pop-tags?        uint8
|   +--rw push-tags!
|   |   +--rw outer-tag
|   |   |   +--rw tag-type      dot1q-tag-type
|   |   |   +--rw vlan-id      vlanid
|   |   +--rw second-tag!
|   |   |   +--rw tag-type      dot1q-tag-type
|   |   |   +--rw vlan-id      vlanid
+--rw egress
|   +--rw dot1q-tag-rewrite {dot1q-tag-rewrites}?
|   +--rw pop-tags?        uint8
|   +--rw push-tags!
|   |   +--rw outer-tag
|   |   |   +--rw tag-type      dot1q-tag-type
|   |   |   +--rw vlan-id      vlanid
|   |   +--rw second-tag!
|   |   |   +--rw tag-type      dot1q-tag-type
|   |   |   +--rw vlan-id      vlanid
+--rw local-traffic-default-encaps!
+--rw outer-tag
|   +--rw tag-type      dot1q-tag-type
|   +--rw vlan-id      vlanid
+--rw second-tag!
|   +--rw tag-type      dot1q-tag-type
|   +--rw vlan-id      vlanid

```

5. VLAN Encapsulation YANG Module

This YANG module augments the 'encapsulation' container defined in `ietf-if-extensions.yang` [I-D.ietf-netmod-intf-ext-yang]. It also contains references to [RFC8343], [RFC7224], and [IEEE802.1Qcp-2018].

```

<CODE BEGINS> file "ietf-if-vlan-encapsulation@2020-07-13.yang"
module ietf-if-vlan-encapsulation {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation";
  prefix if-vlan;

```

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

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

import ieee802-dot1q-types {
  prefix dot1q-types;
  reference
    "IEEE Std 802.1Qcp-2018: IEEE Standard for Local and
    metropolitan area networks -- Bridges and Bridged Networks --
    Amendment 30: YANG Data Model";
}

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

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

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

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

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

  Copyright (c) 2020 IETF Trust and the persons identified as
  authors of the code.  All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject to
  the license terms contained in, the Simplified BSD License set
  forth in Section 4.c of the IETF Trust's Legal Provisions
```

Relating to IETF Documents
(<https://trustee.ietf.org/license-info>).

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

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

```
revision 2020-07-13 {
  description
    "Latest draft revision";
  reference
    "RFC XXXX: Sub-interface VLAN YANG Data Models";
}

augment "/if:interfaces/if:interface/if-ext:encapsulation/"
+ "if-ext:encaps-type" {
  when "derived-from-or-self(..if:type,
    'ianaift:ethernetCsmacd') or
    derived-from-or-self(..if:type,
    'ianaift:ieee8023adLag') or
    derived-from-or-self(..if:type, 'ianaift:l2vlan') or
    derived-from-or-self(..if:type,
    'if-ext:ethSubInterface')" {
    description
      "Applies only to Ethernet-like interfaces and
      sub-interfaces.";
  }
}

description
  "Augment the generic interface encapsulation with basic 802.1Q
  VLAN tag classifications";

case dot1q-vlan {
  container dot1q-vlan {

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

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

sub-interface/interface.

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

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

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

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

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

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

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

    error-message
      "When matching two 802.1Q VLAN tags, the outermost
        (first) tag in the frame MUST be specified and be of
        S-VLAN type and the second tag in the frame must be of
        C-VLAN tag type.";

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

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

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

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

6. Flexible Encapsulation YANG Module

This YANG module augments the 'encapsulation' container defined in ietf-if-extensions.yang [I-D.ietf-netmod-intf-ext-yang]. This YANG module also augments the 'interface' list entry defined in [RFC8343]. It also contains references to [RFC7224], and [IEEE802.1Qcp-2018].

```
<CODE BEGINS> file "ietf-if-flexible-encapsulation@2020-07-13.yang"
module ietf-if-flexible-encapsulation {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-if-flexible-encapsulation";
  prefix if-flex;

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

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

  import ieee802-dot1q-types {
    prefix dot1q-types;
    reference
      "IEEE Std 802.1Qcp-2018: IEEE Standard for Local and
      metropolitan area networks -- Bridges and Bridged Networks --
      Amendment 30: YANG Data Model";
  }

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

```
}

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

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

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

description
  "This YANG module describes interface configuration for flexible
  classification and rewrites of IEEE 802.1Q VLAN tagged Ethernet
  traffic.

  Copyright (c) 2020 IETF Trust and the persons identified as
  authors of the code.  All rights reserved.

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

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

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

revision 2020-07-13 {
  description
    "Latest draft revision";
  reference
    "RFC XXXX: Sub-interface VLAN YANG Data Models";
}

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



```
}

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

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

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

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

  choice match-type {
    mandatory true;

    description
      "Provides a choice of how the frames may be
      matched";

    case default {
      description
        "Default match";

      leaf default {
        type empty;

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

```
    }  
  }  
  
  case untagged {  
    description  
      "Match untagged Ethernet frames only";  
  
    leaf untagged {  
      type empty;  
  
      description  
        "Untagged match.  Matches all untagged traffic.";  
    }  
  }  
  
  case dot1q-priority-tagged {  
    description  
      "Match 802.1Q priority tagged Ethernet frames only";  
  
    container dot1q-priority-tagged {  
      description  
        "802.1Q priority tag match";  
  
      leaf tag-type {  
        type dot1q-types:dot1q-tag-type;  
        mandatory true;  
  
        description  
          "The 802.1Q tag type of matched priority  
          tagged packets";  
      }  
    }  
  }  
  
  case dot1q-vlan-tagged {  
    container dot1q-vlan-tagged {  
      description  
        "Matches VLAN tagged frames";  
  
      container outer-tag {  
        must 'tag-type = "dot1q-types:s-vlan" or '  
          + 'tag-type = "dot1q-types:c-vlan"' {  
  
          error-message  
            "Only C-VLAN and S-VLAN tags can be matched.";  
  
          description  
            "For IEEE 802.1Q interoperability, only C-VLAN and
```

```
        S-VLAN tags can be matched.";
    }

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

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

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

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

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

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

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

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

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

```
    }
  }
}

grouping dot1q-tag-rewrite {
  description
    "Flexible rewrite grouping. Can be either be expressed
    symmetrically, or independently in the ingress and/or egress
    directions.";

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

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

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

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

    description
      "The 802.1Q tags to push on the front of the frame, or
      translate if configured in conjunction with pop-tags.";

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

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

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

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

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

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

```
    + 'tag-type = "dot1q-types:c-vlan"' {  
      error-message  
        "When pushing/rewriting two tags, the outermost tag must  
        be specified and of S-VLAN type and the second outermost  
        tag must be of C-VLAN tag type.";  
  
      description  
        "For IEEE 802.1Q interoperability, when pushing two tags,  
        it is required that the outermost tag exists and is an  
        S-VLAN, and the second outermost tag is a C-VLAN.";  
    }  
  
    presence  
      "In addition to the first tag, also push/rewrite a second  
      VLAN tag.";  
  
    description  
      "The second outermost VLAN tag to push onto the frame.";  
  
    uses dot1q-types:dot1q-tag-classifier-grouping;  
  }  
}  
}  
  
grouping flexible-rewrite {  
  description  
    "Grouping for flexible rewrites of fields in the L2 header.  
  
    Restricted to flexible 802.1Q VLAN tag rewrites, but could be  
    extended to cover rewrites of other fields in the L2 header in  
    future.";  
  
  container dot1q-tag-rewrite {  
    if-feature "dot1q-tag-rewrites";  
  
    description  
      "802.1Q VLAN tag rewrite.  
  
      Translate operations are expressed as a combination of tag  
      push and pop operations. E.g., translating the outer tag is  
      expressed as popping a single tag, and pushing a single tag.  
      802.1Q tags that are translated SHOULD preserve the PCP and  
      DEI fields unless if a different QoS behavior has been  
      specified.";  
    uses dot1q-tag-rewrite;  
  }  
}
```

```
augment "/if:interfaces/if:interface/if-ext:encapsulation/"
+ "if-ext:encaps-type" {
  when "derived-from-or-self(..if:type,
    'ianaift:ethernetCsmacd') or
    derived-from-or-self(..if:type,
    'ianaift:ieee8023adLag') or
    derived-from-or-self(..if:type, 'ianaift:l2vlan') or
    derived-from-or-self(..if:type,
    'if-ext:ethSubInterface')" {

    description
      "Applies only to Ethernet-like interfaces and
      sub-interfaces.";
  }

  description
    "Augment the generic interface encapsulation with flexible
    match and rewrite for VLAN sub-interfaces.";

  case flexible {
    description
      "Flexible encapsulation and rewrite";

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

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

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

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

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

          If a frame could be classified to multiple
          sub-interfaces then they get classified to the
```

```
        sub-interface with the most specific match.  E.g.,
        matching two VLAN tags in the frame is more specific
        than matching the outermost VLAN tag, which is more
        specific than the catch all 'default' match.";

    uses flexible-match;
}

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

    description
        "L2 frame rewrite operations.

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

    choice direction {
        description
            "Whether the rewrite policy is symmetrical or
            asymmetrical.";

        case symmetrical {
            container symmetrical {
                uses flexible-rewrite;

                description
                    "Symmetrical rewrite.  Expressed in the ingress
                    direction, but the reverse operation is applied to
                    egress traffic.

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

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

            description
                "Asymmetrical rewrite.

                Rewrite operations may be specified in only a single
```

```
direction, or different rewrite operations may be
specified in each direction.";

container ingress {
  uses flexible-rewrite;

  description
    "A rewrite operation that only applies to ingress
    traffic.

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

container egress {
  uses flexible-rewrite;

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

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

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

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

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

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

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


conjunction with the IETF L2VPN YANG model [I-D.ietf-bess-l2vpn-yang].

7.1. Layer 3 sub-interfaces with IPv6

This example illustrates two layer sub-interfaces, 'eth0.1' and 'eth0.2', both are child interfaces of the Ethernet interface 'eth0'.

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

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

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces
    xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
    xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
    xmlns:dot1q-types="urn:ieee:std:802.1Q:yang:ieee802-dot1q-types"
    xmlns:if-ext="urn:ietf:params:xml:ns:yang:ietf-if-extensions">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
    </interface>
    <interface>
      <name>eth0.1</name>
      <type>ianaift:l2vlan</type>
      <if-ext:parent-interface>eth0</if-ext:parent-interface>
      <if-ext:encapsulation>
        <dot1q-vlan
          xmlns=
            "urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation">
          <outer-tag>
            <tag-type>dot1q-types:s-vlan</tag-type>
            <vlan-id>10</vlan-id>
          </outer-tag>
          <second-tag>
            <tag-type>dot1q-types:c-vlan</tag-type>
            <vlan-id>20</vlan-id>
          </second-tag>
        </dot1q-vlan>
      </if-ext:encapsulation>
      <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
        <forwarding>true</forwarding>
        <address>
          <ip>2001:db8:10::1</ip>
```

```

        <prefix-length>48</prefix-length>
      </address>
    </ipv6>
  </interface>
  <interface>
    <name>eth0.2</name>
    <type>ianaift:l2vlan</type>
    <if-ext:parent-interface>eth0</if-ext:parent-interface>
    <if-ext:encapsulation>
      <dot1q-vlan
        xmlns=
          "urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation">
        <outer-tag>
          <tag-type>dot1q-types:s-vlan</tag-type>
          <vlan-id>11</vlan-id>
        </outer-tag>
      </dot1q-vlan>
    </if-ext:encapsulation>
    <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
      <forwarding>true</forwarding>
      <address>
        <ip>2001:db8:11::1</ip>
        <prefix-length>48</prefix-length>
      </address>
    </ipv6>
  </interface>
</interfaces>
</config>

```

7.2. Layer 2 sub-interfaces with L2VPN

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

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

```

<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces
    xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"

```

```
xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
xmlns:dot1q-types="urn:ieee:std:802.1Q:yang:ieee802-dot1q-types"
xmlns:if-ext="urn:ietf:params:xml:ns:yang:ietf-if-extensions">
  <interface>
    <name>eth0</name>
    <type>ianaift:ethernetCsmacd</type>
  </interface>
  <interface>
    <name>eth0.3</name>
    <type>ianaift:l2vlan</type>
    <if-ext:parent-interface>eth0</if-ext:parent-interface>
    <if-ext:encapsulation>
      <flexible xmlns=
        "urn:ietf:params:xml:ns:yang:ietf-if-flexible-encapsulation">
        <match>
          <dot1q-vlan-tagged>
            <outer-tag>
              <tag-type>dot1q-types:s-vlan</tag-type>
              <vlan-id>10</vlan-id>
            </outer-tag>
            <second-tag>
              <tag-type>dot1q-types:c-vlan</tag-type>
              <vlan-id>21</vlan-id>
            </second-tag>
          </dot1q-vlan-tagged>
        </match>
        <rewrite>
          <symmetrical>
            <dot1q-tag-rewrite>
              <pop-tags>1</pop-tags>
            </dot1q-tag-rewrite>
          </symmetrical>
        </rewrite>
      </flexible>
    </if-ext:encapsulation>
  </interface>
  <interface>
    <name>eth1</name>
    <type>ianaift:ethernetCsmacd</type>
  </interface>
  <interface>
    <name>eth1.0</name>
    <type>ianaift:l2vlan</type>
    <if-ext:parent-interface>eth0</if-ext:parent-interface>
    <if-ext:encapsulation>
      <flexible xmlns=
        "urn:ietf:params:xml:ns:yang:ietf-if-flexible-encapsulation">
        <match>
```

```

        <dot1q-vlan-tagged>
          <outer-tag>
            <tag-type>dot1q-types:c-vlan</tag-type>
            <vlan-id>50</vlan-id>
          </outer-tag>
        </dot1q-vlan-tagged>
      </match>
    <rewrite>
      <symmetrical>
        <dot1q-tag-rewrite>
          <pop-tags>1</pop-tags>
        </dot1q-tag-rewrite>
      </symmetrical>
    </rewrite>
  </flexible>
</if-ext:encapsulation>
</interface>
</interfaces>
<network-instances
  xmlns="urn:ietf:params:xml:ns:yang:ietf-network-instance">
  <network-instance
    xmlns:l2vpn="urn:ietf:params:xml:ns:yang:ietf-l2vpn">
    <name>p2p-l2-1</name>
    <description>Point to point L2 service</description>
    <l2vpn:type>l2vpn:vpws-instance-type</l2vpn:type>
    <l2vpn:signaling-type>
      l2vpn:ldp-signaling
    </l2vpn:signaling-type>
    <endpoint xmlns="urn:ietf:params:xml:ns:yang:ietf-l2vpn">
      <name>local</name>
      <ac>
        <name>eth0.3</name>
      </ac>
    </endpoint>
    <endpoint xmlns="urn:ietf:params:xml:ns:yang:ietf-l2vpn">
      <name>remote</name>
      <pw>
        <name>pw1</name>
      </pw>
    </endpoint>
    <vsi-root>
    </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 Benoit Claise, John Messenger, Glenn Parsons, and Dan Romascanu for their help progressing this draft.

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

9. ChangeLog

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

9.1. WG version -07 and -06

- o Apply markups from WG last call.

9.2. WG version -05

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

9.3. WG version -04

- o Added examples

9.4. WG version -03

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

9.5. WG version -02

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

9.6. WG version -01

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

9.7. Version -04

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

9.8. Version -03

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

10. IANA Considerations

10.1. YANG Module Registrations

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

The ietf-if-vlan-encapsulation module:

Name: ietf-if-vlan-encapsulation

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

Prefix: if-vlan

Reference: [RFCXXXX]

The ietf-if-flexible-encapsulation module:

Name: ietf-if-flexible-encapsulation

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

Prefix: if-flex

Reference: [RFCXXXX]

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

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

Registrant Contact: The IESG.

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

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

Registrant Contact: The IESG.

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

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. ietf-if-vlan-encapsulation.yang

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

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

11.2. ietf-if-flexible-encapsulation.yang

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

- o default
- o untagged
- o dot1q-priority-tagged
- o dot1q-priority-tagged/tag-type
- o dot1q-vlan-tagged/outer-tag/vlan-type

- o dot1q-vlan-tagged/outer-tag/vlan-id
- o dot1q-vlan-tagged/second-tag/vlan-type
- o 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:

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

- o 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:

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

12. References

12.1. Normative References

- [I-D.ietf-netmod-intf-ext-yang]
Wilton, R., Ball, D., tapsingh@cisco.com, t., and S. Sivaraj, "Common Interface Extension YANG Data Models", draft-ietf-netmod-intf-ext-yang-08 (work in progress), November 2019.
- [IEEE802.1Qcp-2018]
Holness, M., "IEEE Std 802.1Qcp-2018: IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks -- Amendment 30: YANG Data Model", 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>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.

- [RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", RFC 7224, DOI 10.17487/RFC7224, May 2014, <<https://www.rfc-editor.org/info/rfc7224>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.
- [RFC8344] Bjorklund, M., "A YANG Data Model for IP Management", RFC 8344, DOI 10.17487/RFC8344, March 2018, <<https://www.rfc-editor.org/info/rfc8344>>.

12.2. Informative References

- [I-D.ietf-bess-l2vpn-yang] Shah, H., Brissette, P., Chen, I., Hussain, I., Wen, B., and K. Tiruveedhula, "YANG Data Model for MPLS-based L2VPN", draft-ietf-bess-l2vpn-yang-10 (work in progress), July 2019.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<https://www.rfc-editor.org/info/rfc2460>>.
- [RFC4448] Martini, L., Ed., Rosen, E., El-Aawar, N., and G. Heron, "Encapsulation Methods for Transport of Ethernet over MPLS Networks", RFC 4448, DOI 10.17487/RFC4448, April 2006, <<https://www.rfc-editor.org/info/rfc4448>>.
- [RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <<https://www.rfc-editor.org/info/rfc4761>>.
- [RFC4762] Lasserre, M., Ed. and V. Kompella, Ed., "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling", RFC 4762, DOI 10.17487/RFC4762, January 2007, <<https://www.rfc-editor.org/info/rfc4762>>.

- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<https://www.rfc-editor.org/info/rfc6536>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.

Appendix A. Comparison with the IEEE 802.1Q Configuration Model

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

A.1. Sub-interface based configuration model overview

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

- o 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.
- o 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.
- o 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.

- o 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.
- o 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:

- o 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.
- o 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.
- o Classification of traffic to a VLAN bridge component is based only on the Ethernet interface that it arrived on.
- o 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.
- o 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
Updates: 8340 (if approved)
Intended status: Standards Track
Expires: June 12, 2020

A. Bierman
YumaWorks
M. Bjorklund
Cisco
K. Watsen
Watsen Networks
December 10, 2019

YANG Data Structure Extensions
draft-ietf-netmod-yang-data-ext-05

Abstract

This document describes YANG mechanisms for defining abstract data structures with YANG.

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 June 12, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
1.1. Terminology	3
1.1.1. NMDA	3
1.1.2. YANG	3
2. Definitions	4
3. YANG Data Structures in YANG Tree Diagrams	5
4. YANG Data Structure Extensions Module	5
5. IANA Considerations	10
5.1. YANG Module Registry	10
6. Security Considerations	10
7. References	10
7.1. Normative References	10
7.2. Informative References	11
Appendix A. Examples	11
A.1. "structure" Example	11
A.2. "augment-structure" Example	13
A.3. XML Encoding Example	13
A.4. JSON Encoding Example	14
A.5. "structure" example that defines a non-top-level structure	14
Authors' Addresses	15

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] was defined for this purpose but it is limited in its functionality.

The intended use of the "yang-data" extension was to model all or part of a protocol message, such as the "errors" definition in the YANG module "ietf-restconf" [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 "structure", which is similar to but more flexible than the "yang-data" extension from [RFC8040]. There is no assumption that a YANG data structure can only be used as a top-level abstraction, and it may also be nested within some other data structure.

This document also defines a new YANG extension statement called "augment-structure", which allows abstract data structures to be

augmented from external modules, similarly 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 "structure" extension.

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 used within this document:

- o YANG data structure: A data structure defined with the "structure" statement.

1.1.1. NMDA

The following terms are defined in the Network Management Datastore Architecture (NMDA) [RFC8342], and are not redefined here:

- o configuration
- o operational state

1.1.2. YANG

The following terms are defined in [RFC7950]:

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

2. Definitions

A YANG data structure is defined with the "structure" extension statement, defined in the YANG module "ietf-yang-structure-ext". The argument to the "structure" extension statement is the name of the data structure. The data structures 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 data structures defined with the "structure" statement cannot have the same name as sibling nodes from regular YANG data definition statements or other "structure" statements in the same YANG module.

This does not mean a YANG data structure, once defined, has to be used as a top-level protocol message or other top-level data structure.

A YANG data structure is encoded in the same way as an "anydata" node. This means that the name of the structure is encoded as a "container", with the instantiated children encoded as child nodes to this node. For example, this structure:

```
module example-errors {  
  ...  
  
  sx:structure my-error {  
    leaf error-number {  
      type int;  
    }  
  }  
}
```

can be encoded in JSON as:

```
"example-errors:my-error": {  
  "error-number": 131  
}
```

3. YANG Data Structures in YANG Tree Diagrams

A YANG data structure can be printed in a YANG Tree Diagram [RFC8340]. This document updates RFC 8340 by defining two new sections in the tree diagram for a module:

1. YANG data structures, offset by two spaces and identified by the keyword "structure" followed by the name of the YANG data structure and a colon (":") character.
2. YANG data structure augmentations, offset by 2 spaces and identified by the keyword "augment-structure" followed by the augment target structure name and a colon (":") character.

The new sections, including spaces conventions is:

```
structure <structure-name>:
  +---<node>
    +---<node>
      |   +---<node>
      +---<node>
structure <structure-name>:
  +---<node>

augment-structure <structure-name>:
  +---<node>
    +---<node>
      |   +---<node>
      +---<node>
augment-structure <structure-name>:
  +---<node>
```

Nodes in YANG data structures are printed according to the rules defined in section 2.6 in [RFC8340].

4. YANG Data Structure Extensions Module

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

<CODE BEGINS> file "ietf-yang-structure-ext@2019-03-07.yang"

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

  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:kent+ietf@watsen.net>";
description
  "This module contains conceptual YANG specifications for defining
  abstract data structures.

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

  Copyright (c) 2019 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

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

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

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

revision 2019-03-07 {
  description
    "Initial revision.";
  // RFC Ed.: replace XXXX with RFC number and remove this note
  reference
    "RFC XXXX: YANG Structure Extensions.";
}
```

```
extension structure {  
  argument name {  
    yin-element true;  
  }  
  description  
    "This extension is used to specify a YANG data structure that  
    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 'structure' extension statement specify the generic  
    syntax for the specific YANG data structure, whose name is the  
    argument of the 'structure' 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, if applicable.

The mandatory 'name' parameter value identifies the YANG data structure that is being defined.

This extension is only valid as a top-level statement, i.e., given as a sub-statement to 'module' or 'submodule'.

The sub-statements of this extension MUST follow the ABNF rules below, where the rules are defined in RFC 7950:

```
*must-stmt  
[status-stmt]  
[description-stmt]  
[reference-stmt]  
*(typedef-stmt / grouping-stmt)  
*data-def-stmt
```

A YANG data structure defined with this extension statement is encoded in the same way as an 'anydata' node. This means that the name of the structure is encoded as a 'container', with the instantiated child statements encoded as child nodes to this node.

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 structure.

The XPath document element is the extension statement itself, such that the child nodes of the document element 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 'structure' extension statement.

- The list-stmt is not required to have a key-stmt defined.
- The config-stmt is ignored if present.

```
    ";\n  }
```

```
extension augment-structure {\n  argument path {\n    yin-element true;\n  }\n  description
```

```
    "This extension is used to specify an augmentation to YANG data\n    structure defined with the 'structure' statement. It is\n    intended to describe hierarchical data independent of protocol\n    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 structure, 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, where the first node in the absolute-schema-nodeid string identifies the YANG data structure to augment, and the rest of the nodes in the string identifies the node within the YANG structure to augment.

This extension is only valid as a top-level statement, i.e., given as a sub-statement to 'module' or 'submodule'.

The sub-statements of this extension MUST follow the ABNF rules below, where the rules are defined in RFC 7950:


```
[status-stmt]
[description-stmt]
[reference-stmt]
1*(data-def-stmt / case-stmt)
```

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 element is the augmented extension statement itself, such that the child nodes of the document element are represented by the data-def-stmt sub-statements within the augmented 'structure' statement.

The context node of the 'augment-structure' 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 an 'augment-structure' extension statement.

- The list-stmt is not required to have a key-stmt defined.
- The config-stmt is ignored if present.

Example:

```
module foo {
  import ietf-yang-structure-ext { prefix sx; }

  sx:structure foo-data {
    container foo-con { }
  }
}

module bar {
  import ietf-yang-structure-ext { prefix sx; }
```

```
import foo { prefix foo; }

sx:augment-structure /foo:foo-data/foo:foo-con {
  leaf add-leaf1 { type int32; }
  leaf add-leaf2 { type string; }
}

";
}
}
```

<CODE ENDS>

5. IANA Considerations

5.1. 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-structure-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-structure-ext
namespace: urn:ietf:params:xml:ns:yang:ietf-yang-structure-ext
prefix: sx
// RFC Ed.: replace XXXX with RFC number and remove this note
reference: RFC XXXX

6. Security Considerations

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

7. References

7.1. Normative References

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

- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [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>>.

7.2. Informative References

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

Appendix A. Examples

A.1. "structure" Example

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

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

  import ietf-yang-structure-ext {
    prefix sx;
  }

  sx:structure 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";
      }
    }
  }
}
```

Below is the tree diagram of this module.

```
module: example-module

  structure address-book:
    +-- address* [last first]
      +-- last      string
      +-- first     string
      +-- street?   string
      +-- city?     string
      +-- state?    string
```

A.2. "augment-structure" Example

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

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

  import ietf-yang-structure-ext {
    prefix sx;
  }
  import example-module {
    prefix exm;
  }

  sx:augment-structure "/exm:address-book/exm:address" {
    leaf county {
      type string;
      description "County name";
    }
    leaf zipcode {
      type string;
      description "Postal zipcode";
    }
  }
}
```

Below is the tree diagram of this module.

```
module: example-module-aug

  augment-structure /exm:address-book/exm:address:
    +-- county?   string
    +-- zipcode?  string
```

A.3. XML Encoding Example

This example shows how an address book can be encoded in XML:

```
<address-book xmlns="urn:example:example-module">
  <address>
    <last>Flintstone</last>
    <first>Fred</first>
    <street>301 Cobblestone Way</street>
    <city>Bedrock</city>
    <zipcode xmlns="urn:example:example-module-aug">70777</zipcode>
  </address>
  <address>
    <last>Root</last>
    <first>Charlie</first>
    <street>4711 Cobblestone Way</street>
    <city>Bedrock</city>
    <zipcode xmlns="urn:example:example-module-aug">70777</zipcode>
  </address>
</address-book>
```

A.4. JSON Encoding Example

This example shows how an address book can be encoded in JSON:

```
"example-module:address-book": {
  "address": [
    {
      "city": "Bedrock",
      "example-module-aug:zipcode": "70777",
      "first": "Fred",
      "last": "Flintstone",
      "street": "301 Cobblestone Way"
    },
    {
      "city": "Bedrock",
      "example-module-aug:zipcode": "70777",
      "first": "Charlie",
      "last": "Root",
      "street": "4711 Cobblestone Way"
    }
  ]
}
```

A.5. "structure" example that defines a non-top-level structure

The following example defines a data structure with error information, that can be included in an `<error-info>` element in an `<rpc-error>`.

```
module example-error-info {  
  yang-version 1.1;  
  namespace "urn:example:example-error-info";  
  prefix exei;  
  
  import ietf-yang-structure-ext {  
    prefix sx;  
  }  
  
  sx:structure my-example-error-info {  
    leaf error-code {  
      type uint32;  
    }  
  }  
}
```

The example below shows how this structure can be used in an `<rpc-error>`.

```
<rpc-reply message-id="101"  
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">  
  <rpc-error>  
    <error-type>protocol</error-type>  
    <error-tag>operation-failed</error-tag>  
    <error-severity>error</error-severity>  
    <error-info>  
      <my-example-error-info  
        xmlns="urn:example:example-error-info">  
          <error-code>42</error-code>  
        </my-example-error-info>  
      </error-info>  
    </rpc-error>  
  </rpc-reply>
```

Authors' Addresses

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Martin Bjorklund
Cisco

Email: mbj@tail-f.com

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.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.

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 19, 2019.

Copyright Notice

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

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

Table of Contents

1. Introduction	2
2. Requirements Language	3
3. Goals	3
3.1. Automated Folding of Long Lines in Artwork	3
3.2. Automated Reconstitution of Original Artwork	4
4. Limitations	4
4.1. Not Recommended for Graphical Artwork	4
4.2. Doesn't Work as Well as Format-Specific Options	4
5. Folded Structure	5
5.1. Header	5
5.2. Body	5
6. Algorithm	6
6.1. Automated Folding	6
6.1.1. Manual Folding	7
6.2. Automated Unfolding	7
7. Considerations for xml2rfc v3	8
8. Examples	8
8.1. Simple Example Showing Boundary Conditions	8
8.2. Example Showing Multiple Wraps of a Single Line	9
8.3. Example With Native Backslash	9
8.4. Example With Native Whitespace	9
8.5. Example of Manual Wrapping	9
9. Security Considerations	10
10. IANA Considerations	10
11. References	10
11.1. Normative References	10
11.2. Informative References	10
Appendix A. POSIX Shell Script	12
Acknowledgements	16
Authors' Addresses	17

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) =====

```
123456789012345678901234567890123456789012345678901234567890123456
1234567890123456789012345678901234567890123456789012345678901234567
12345678901234567890123456789012345678901234567890123456789012345678
123456789012345678901234567890123456789012345678901234567890123456789
12345678901234567890123456789012345678901234567890123456789012345678\
\90
12345678901234567890123456789012345678901234567890123456789012345678\
\901
12345678901234567890123456789012345678901234567890123456789012345678\
\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) =====

```
1234567890123456789012345678901234567890123456789012345678\
\901234567890123456789012345678901234567890123456789012345\
\678901234567890123456789012345678901234567890123456789012\
\345678901234567890123456789012345678901234567890123456789\
\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
pcgrep -M "\\n[\ ]*\\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" < $infile\
\ >> $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.

Copyright Notice

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

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

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

Table of Contents

1. Terminology	2
2. Introduction	3
2.1. Use Cases	3
2.1.1. Use Case 1: Early Documentation of Server Capabilites	3
2.1.2. Use Case 2: Preloading Data	4
2.1.3. Use Case 3: Dcoumenting Factory Default Settings . .	4
3. Instance Data File Format	5
4. Data Life cycle	8
5. Delivery of Instance Data	9
6. YANG Model	9
7. Security Considerations	11
8. IANA Considerations	11
9. References	11
9.1. Normative References	11
9.2. Informative References	11
Appendix A. Open Issues	12
Appendix B. Changes between revisions	13
Authors' Addresses	14

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 where Yang based instance data is needed.

2.1.1. Use Case 1: Early Documentation of Server Capabilities

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

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

While it is good practice to allow a client to query these capabilities 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 capabilities.

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

Most server-capabilities are relatively stable and change only during upgrade or due to licensing or addition or removal of HW. They are usually defined by a vendor in design time, before the product is released. It is 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 capabilities 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 configuration.

2.1.3. Use Case 3: Documenting 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 returned 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 configuration 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 constraints 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:

- o Name of the instance data set

Meta data SHOULD include:

- o Revision date of the instance data set
- o 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

- o 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.
- o 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.
- o 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.
- o 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

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

v03 - v04

- o Updated changelog for v02-v03

v02 - v03

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

v01 - v02

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

v00 - v01

- o Redefined using yang-data-ext
- o 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: January 3, 2019

N. Sambo
P. Castoldi
Scuola Superiore Sant'Anna
G. Fioccola
Telecom Italia
F. Cugini
CNIT
H. Song
T. Zhou
Huawei
July 2, 2018

YANG model for finite state machine
draft-sambo-netmod-yang-fsm-03

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

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.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Conventions used in this document	3
3. Terminology	4
4. Example of application	4
4.1. Pre-programming resiliency schemes in EONs	4
4.2. Deploying Dynamic Probes for Programmable Network Telemetry	8
4.3. IP Performance Measurements on multipoint-to-multipoint large Networks	10
5. YANG for finite state machine (FSM)	11
6. Implementation of the pre-programming resiliency schemes in EONs	14
7. Appendix	15
7.1. YANG model for FSM - Tree	15
7.2. YANG model for FSM - Code	16
7.3. Example of values for the YANG model	28
8. Acknowledgements	29
9. Other Contributors	29
10. Security Considerations	30
11. IANA Considerations	30
12. References	30
12.1. Normative References	30
12.2. Informative References	30
Authors' Addresses	31

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 programmed 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 on) 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.

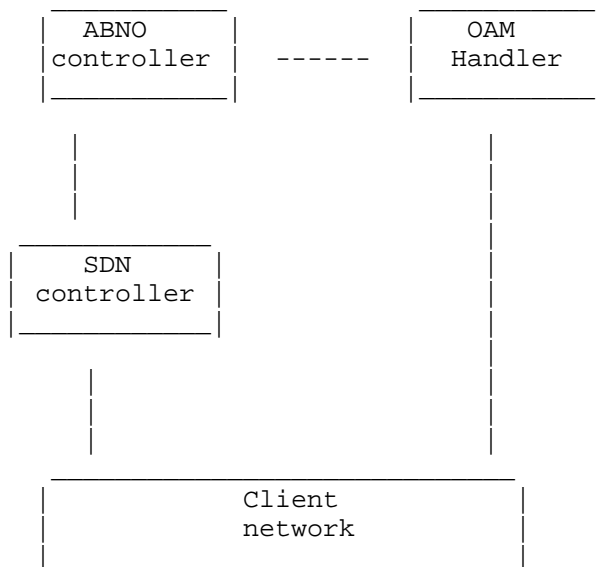


Figure 1: Assumed ABNO functional modules

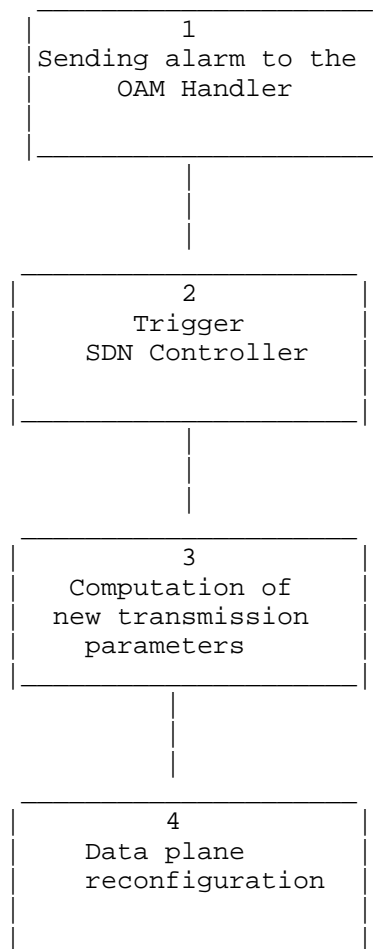


Figure 2: Flow chart of the expected state-of-the-art approach

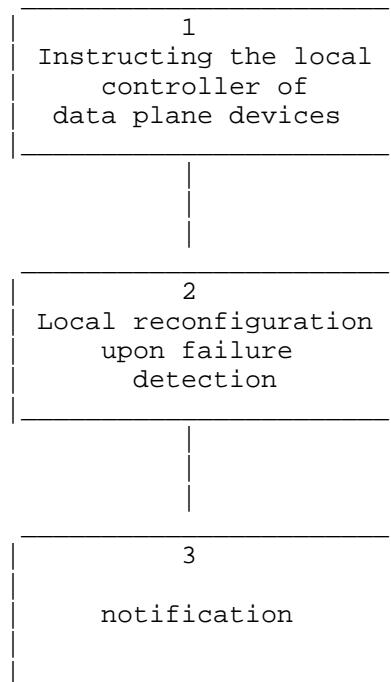


Figure 3: Flow chart of the approach exploiting YANG models in this draft

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.

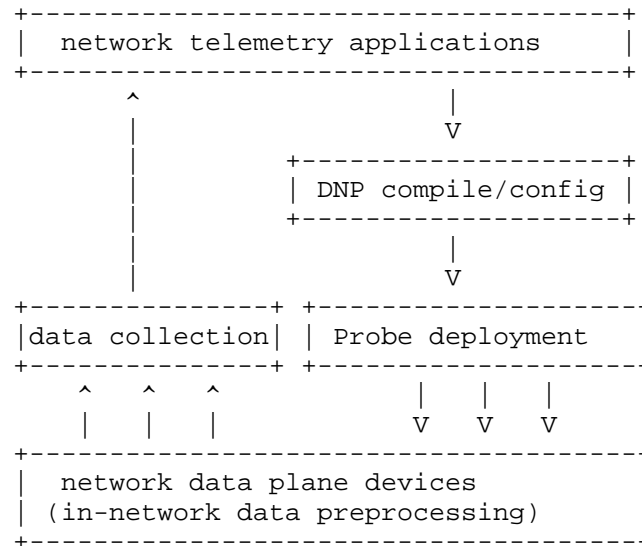


Figure 4: Deploy dynamic network probes using YANG FSM models

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).

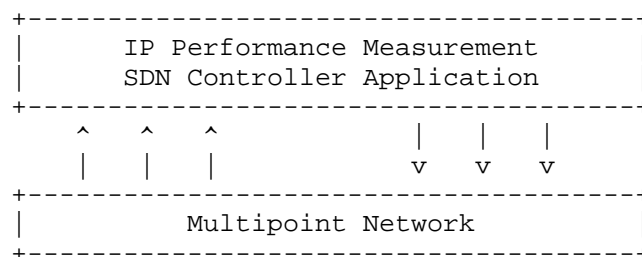


Figure 5: Feedback mechanism on multipoint-to-multipoint large Networks

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 9×10^{-4} . A transition from this state can be triggered by the event with <name>=BER_CHANGE and <filter-type>= 9×10^{-4} , thus expressing a change of the pre-FEC BER above the threshold. In case the pre-FEC

BER exceeds 9×10^{-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
Telecom Italia
Via Reiss Romoli, 274
Torino 10148
Italy

Email: giuseppe.fioccola@telecomitalia.it

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

NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 22, 2019

N. Sambo
P. Castoldi
Scuola Superiore Sant'Anna
G. Fioccola
Huawei Technologies
F. Cugini
CNIT
H. Song
T. Zhou
Huawei
May 21, 2019

YANG model for finite state machine
draft-sambo-netmod-yang-fsm-05

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; iv) for re-routing in optical networks.

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 November 22, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Conventions used in this document	4
3. Terminology	4
4. Example of application	4
4.1. Pre-programming resiliency schemes in EONs	4
4.2. Deploying Dynamic Probes for Programmable Network Telemetry	8
4.3. IP Performance Measurements on multipoint-to-multipoint large Networks	10
4.4. Re-routing in optical networks	11
5. YANG for finite state machine (FSM)	12
6. Implementation of the pre-programming resiliency schemes in EONs	15
7. Appendix	16
7.1. YANG model for FSM - Tree	16
7.2. YANG model for FSM - Code	16
7.3. Example of values for the YANG model	29
8. Acknowledgements	30
9. Other Contributors	30
10. Security Considerations	31
11. IANA Considerations	31

12. References	31
12.1. Normative References	31
12.2. Informative References	31
Authors' Addresses	32

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.ietf-ippm-multipoint-alt-mark]). Finally, a use case is related to enable automatic local reconfiguration of a backup route in optical networks.

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 losing 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 on) 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.

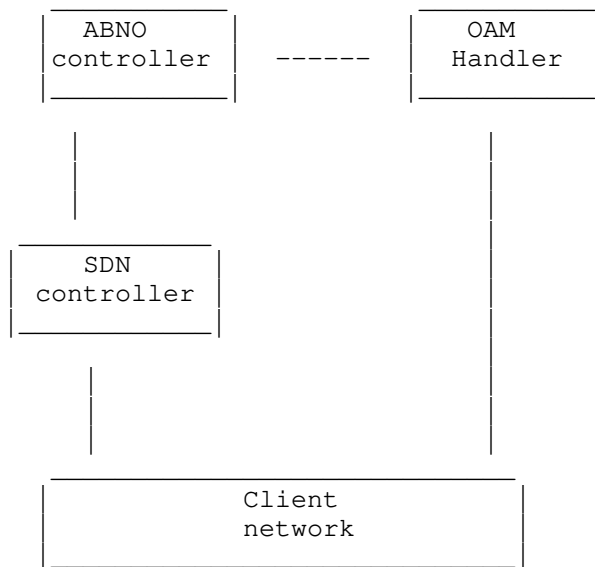


Figure 1: Assumed ABNO functional modules

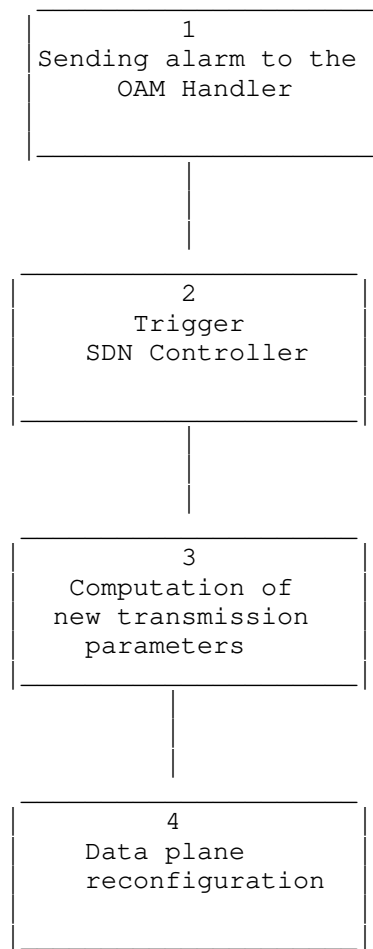


Figure 2: Flow chart of the expected state-of-the-art approach

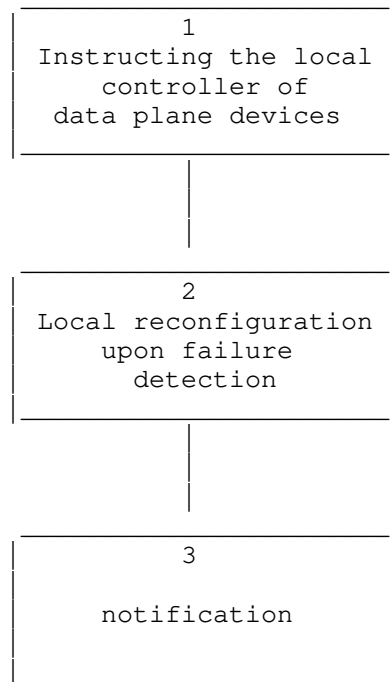


Figure 3: Flow chart of the approach exploiting YANG models in this draft

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.

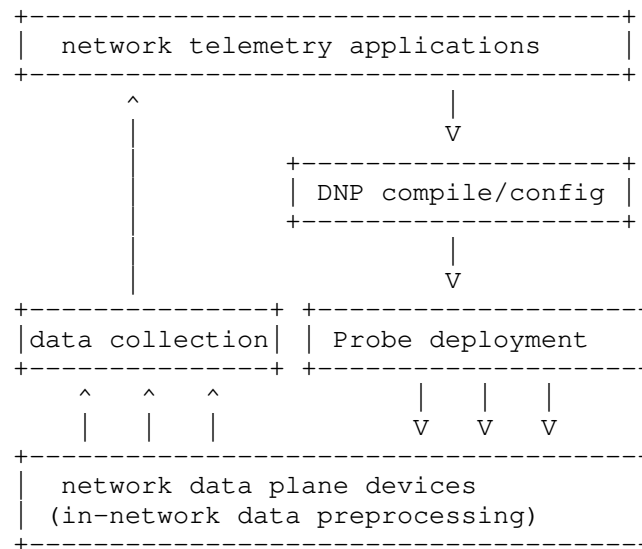


Figure 4: Deploy dynamic network probes using YANG FSM models

Many DNP's 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).

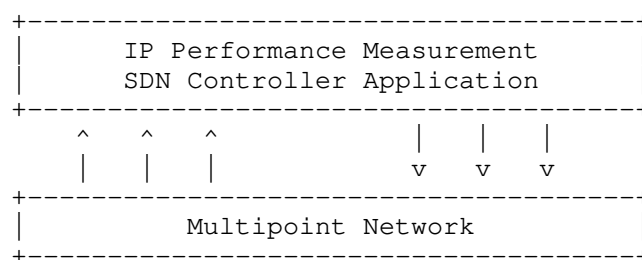


Figure 5: Feedback mechanism on multipoint-to-multipoint large Networks

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 RFC8321 [RFC8321] and [I-D.ietf-ippm-multipoint-alt-mark].

This technique can be applied to point-to-point flows but also to multipoint.to-multipoint flows (see RFC8321 [RFC8321] and [I-D.ietf-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.ietf-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 RFC8321 [RFC8321].

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.

4.4. Re-routing in optical networks

FSM can be also applied for rerouting purposes as dynamic restoration in optical networks. In such a use case all the nodes along the backup route of a media channel should hold a FSM indicating the reconfigurations to be performed in case that media channel is rerouted along the backup path. Note that resources along the backup route are not reserved neither configured during a normal operation of the network as instead it happens for protection.

The proposed approach aims at achieving a sort of hybrid centralized/distributed rerouting solution applicable to all the networks controlled with NETCONF. More specifically, the decision of the backup route is taken centrally by the SDN controller, but it is acted in a distributed way through FSM when a problem appears.

In particular, the transmitter, the receiver, and the ROADMs along the backup route have installed in their agent a FSM including a "re-routing" state, associated to a specific media channel. For the transmitter and receiver, this state is associated to the transmission parameters of the backup media channel: e.g., modulation format, central frequency, FEC, and so on. Regarding the ROADMs, the "re-routing" state is associated to the cross connections of the backup route.

When a link failure occurs (e.g., fiber cut), the receiver reveals a loss of light, which triggers the transition to the "re-routing" state. Thus, the receiver can set itself to the new transmission parameters. Moreover, similarly to the message sent over a control channel to the transmitter in Sec. 4.1, the receiver sends a message to the transmitter and the backup Reconfigurable Add-Drop Multiplexer (ROADMs). The task of this message is to simply trigger a state transition to the "re-routing" state so that each backup ROADM and the transmitter can apply the proper reconfigurations. This way, the SDN controller is not interrogated during the failure and the recovery can be speeded up. After reconfigurations, the SDN controller can be notified and the SDN controller can tear down the primary path. Backup routes can be reprogrammed based on the network states evolutions (e.g., link and spectrum occupancy).

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 9×10^{-4} . A transition from this state can be triggered by the event with <name>=BER_CHANGE and <filter-type>= 9×10^{-4} , thus expressing a change of the pre-FEC BER above the threshold. In case the pre-FEC BER exceeds 9×10^{-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 occurring 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., Lapukhov, P., and r. remy@barefootnetworks.com, "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements-03 (work in progress), March 2017.
- [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-multipoint-alt-mark]
Fioccola, G., Cociglio, M., Sapio, A., and R. Sisto,
"Multipoint Alternate Marking method for passive and
hybrid performance monitoring", draft-ietf-ippm-
multipoint-alt-mark-01 (work in progress), March 2019.
- [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.
- [RFC8321] Fioccola, G., Ed., Capello, A., Cociglio, M., Castaldelli,
L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi,
"Alternate-Marking Method for Passive and Hybrid
Performance Monitoring", RFC 8321, DOI 10.17487/RFC8321,
January 2018, <<https://www.rfc-editor.org/info/rfc8321>>.

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: May 27, 2019

J. Clarke, Ed.
Cisco Systems, Inc.
November 23, 2018

YANG Module Versioning Requirements
draft-verdt-netmod-yang-versioning-reqs-02

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 May 27, 2019.

Copyright Notice

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

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

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

Table of Contents

1. Introduction	2
2. Background	2
2.1. Striving for model perfection	3
2.2. Some YANG Modules Are Not Backwards-Compatible	3
2.3. Non-Backwards-Compatible Errors	4
2.4. No way to easily decide whether a change is Backwards-Compatible	4
2.5. No good way to specify which module revision to import	5
2.6. Early Warning about Removal	6
2.7. Clear Indication of Node Support	6
3. Terminology and Conventions	7
4. The Problem Statement	7
5. Requirements of a YANG Versioning Solution	9
6. Contributors	11
7. Acknowledgments	11
8. Security Considerations	11
9. IANA Considerations	12
10. References	12
10.1. Normative References	12
10.2. Informative References	12
Author's Address	12

1. Introduction

This requirements document initially considers some of the existing YANG module update rules, then describes the problems that arise due to those rules embracing strict backwards compatibility, and finally defines requirements on any solution that may be designed to solve these problems by providing an alternative YANG versioning strategy.

2. Background

The YANG data modeling language [RFC7950] specifies strict rules for updating YANG modules (see section 11 "Updating a Module"). Citing a few of the relevant rules:

1. "As experience is gained with a module, it may be desirable to revise that module. However, changes to published modules are not allowed if they have any potential to cause interoperability problems between a client using an original specification and a server using an updated specification."

2. "Note that definitions contained in a module are available to be imported by any other module and are referenced in "import" statements via the module name. Thus, a module name MUST NOT be changed. Furthermore, the "namespace" statement MUST NOT be changed, since all XML elements are qualified by the namespace."
3. "Otherwise, if the semantics of any previous definition are changed (i.e., if a non-editorial change is made to any definition other than those specifically allowed above), then this MUST be achieved by a new definition with a new identifier."
4. "deprecated indicates an obsolete definition, but it permits new/continued implementation in order to foster interoperability with older/existing implementations."

The rules described above, along with other similar rules, causes various problems, as described in the following sections:

2.1. Striving for model perfection

The points made above lead to the logical conclusion that the standardized YANG modules have to be perfect on day one (at least the structure and meaning), which in turn might explain why IETF YANG modules take so long to standardize. Shooting for perfection is obviously a noble goal, but if the perfect standard comes too late, it doesn't help the industry.

2.2. Some YANG Modules Are Not Backwards-Compatible

As we learn from our mistakes, we're going to face more and more non-backwards-compatible YANG modules. An example is the YANG data model for L3VPN service delivery [RFC8049], which, based on implementation experience, has been updated in a non-backwards-compatible way by [RFC8299].

While Standards Development Organization (SDO) YANG modules are obviously better for the industry, we must recognize that many YANG modules are actually generated YANG modules (for example, from internal databases), which is sometimes the case for vendor modules [RFC8199]. From time to time, the new YANG modules are not backwards-compatible.

Old module parts that are no longer needed, no longer supported, or are not used by consumers need to be removed from modules. It is often hard to decide which parts are no longer needed/used; still the need and practice of removing old parts exist. While it is rare in standard modules it is more common in vendor YANG modules where the usage of modules is more controlled.

The problems described in Section 2.7 may also result in incompatible changes.

In such cases, it would be better to indicate how backwards-compatible a given YANG module actually is.

As modules are sometimes updated in an incompatible way the current assumption that once a YANG module is defined all further revisions can be freely used as they are compatible is not valid.

2.3. Non-Backwards-Compatible Errors

Sometimes small errors force us to make non-backwards-compatible updates. As an example imagine that we have a string with a complex pattern (e.g., an IP address). Let's assume the initial pattern incorrectly allows IP addresses to start with 355. In the next version this is corrected to disallow addresses starting with 355. Formally this is a non-backwards-compatible change as the value space of the string is decreased. In reality an IP address and the implementation behind it was never capable of handling an address starting with 355. So practically this is a backwards-compatible change, just like a correction of the description statement. Current YANG rules are ambiguous as to whether non-backwards-compatible bug fixes are allowed without also requiring a module name change.

2.4. No way to easily decide whether a change is Backwards-Compatible

A management system, SDN controller, or any other user of a module should be capable of easily determining the compatibility between two module versions. Higher level logic for a network function, something that cannot be implemented in a purely model driven way, is always dependent on a specific version of the module. If the client finds that the module has been updated on the network node, it has to decide if it tries to handle it as it handled the previous version of the model or if it just stops, to avoid problems. To make this decision the client needs to know if the module was updated in a backwards-compatible way or not.

This is not possible to decide today because of the following:

- o 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.

- o Problems with the deprecated and obsolete status statement, Section 2.7
- o 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

- o If corrections are made to MOD-A these would not have any effect as the import's revision date would still point to the uncorrected earlier YANG module revision.
- o 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.
- o 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.
- o Tooling/SW is often not prepared to handle multiple revisions of the same YANG module.

If the import revision-date is not specified

- o 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.

- o 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.

- o "deprecated" indicates an obsolete definition, but it permits new/continued implementation in order to foster interoperability with older/existing implementations.
- o "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:

- o YANG module revision: An instance of a YANG module, with no implied ordering or backwards compatibility between different revisions of the same module."
- o 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.
- o 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:

- o 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.
- o 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:

- o 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.
- o 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.
- o 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 backwards-compatible enhancements and bug fixes, as well as non-backwards-compatible bug fixes in non-latest-release modules.
2. Requirements related to identifying changes between different module revisions:
 - 2.1 Readers of modules, and tools that use modules, MUST be able to determine whether changes between two revisions of a module constitute a backwards-compatible or non-backwards-compatible version change. In addition, it MAY be helpful to identify whether changes represent bug fixes, new functionality, or both.

- 2.2 A mechanism SHOULD be defined to determine whether data nodes between two arbitrary YANG module revisions have (i) not changed, (ii) changed in a backwards-compatible way, (iii) changed in a non-backwards-compatible way.
3. Requirements related to supporting existing clients in a backwards-compatible way:
 - 3.1 The solution MUST provide a mechanism to allow servers to support existing clients in a backwards-compatible way.
 - 3.2 The solution MUST provide a mechanism to support clients that expect an older version of a given module when the current version has had non-backwards-compatible changes.
 - 3.3 Clients are expected to be able to handle unexpected instance data resulting from backwards-compatible changes.
4. Requirements related to managing and documenting the life cycle of data nodes:
 - 4.1 A mechanism is REQUIRED to allow a client to determine whether deprecated nodes are implemented by the server.
 - 4.2 If a data node is deprecated or obsolete then it MUST be possible to document in the YANG module what alternatives exist, the reason for the status change, or any other status related information.
 - 4.3 A mechanism is REQUIRED to indicate that certain definitions in a YANG module will become status obsolete in future revisions but definitions marked as such MUST still be implemented by compliant servers.
5. Requirements related to documentation and education:
 - 5.1 The solution MUST provide guidance to model authors and clients on how to use the new YANG versioning scheme.
 - 5.2 The solution is REQUIRED to describe how to transition from the existing YANG 1.0/1.1 versioning scheme to the new scheme.
 - 5.3 The solution MUST describe how the versioning scheme affects the interpretation of instance data and references to instance data, for which the schema definition has been updated in a non-backwards-compatible way.

6. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The following people are members of that design team and have contributed to defining the problem and specifying the requirements:

- o Balazs Lengyel
- o Benoit Claise
- o Ebben Aries
- o Jason Sterne
- o Joe Clarke
- o Juergen Schoenwaelder
- o Mahesh Jethanandani
- o Michael (Wangzitao)
- o Qin Wu
- o Reshad Rahman
- o Rob Wilton

7. Acknowledgments

The design team would like to thank Christian Hopps and Vladimir Vassilev for their feedback and perspectives in shaping and fine tuning the versioning requirements.

One of the inspirations for solving the YANG module versioning comes from OpenConfig. The authors would like to thank Anees Shaikh and Rob Shakir for their helpful input.

8. Security Considerations

The document does not define any new protocol or data model. There is no security impact.

9. IANA Considerations

None

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.

10.2. Informative References

- [RFC8049] Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8049, DOI 10.17487/RFC8049, February 2017, <<https://www.rfc-editor.org/info/rfc8049>>.
- [RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <<https://www.rfc-editor.org/info/rfc8199>>.
- [RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <<https://www.rfc-editor.org/info/rfc8299>>.

Author's Address

Joe Clarke (editor)
Cisco Systems, Inc.
7200-12 Kit Creek Rd
Research Triangle Park, North Carolina
United States of America

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2019

G. Zheng
M. Wang
B. Wu
Huawei
July 1, 2018

Yang data model for Terminal Access Controller Access Control System
Plus
draft-zheng-netmod-tacacs-yang-01

Abstract

This document describes a data model of Terminal Access Controller Access Control System Plus (TACACS+).

The YANG data model in this document conforms 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 January 2, 2019.

Copyright Notice

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

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

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

Table of Contents

1. Introduction	2
2. Conventions used in this document	2
2.1. Tree Diagrams	3
3. Problem Statement	3
4. Design of the Data Model	3
4.1. TACACS+ Modules Overview	4
5. TACACS+ Module	8
6. Security Considerations	30
7. IANA Considerations	30
8. Normative References	31
Authors' Addresses	32

1. Introduction

This document describes a data model of Terminal Access Controller Access Control System Plus (TACACS+). TACACS+ provides Device Administration for routers, network access servers and other networked computing devices via one or more centralized servers. Various TACACS+ clients and servers have been widely deployed.

This document defines a YANG [RFC7950] data model for TACACS+ draft-ietf-opsawg-tacacs-10 implementation and identification of some common properties within a device containing a Network Configuration Protocol (NETCONF) server. Devices that are managed by NETCONF and perhaps other mechanisms have common properties that need to be configured and monitored in a standard way.

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

2. Conventions used in this document

The keywords "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 BCP14, [RFC2119], [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined in [RFC6241] and are used in this specification:

- o client

- o configuration data
- o server
- o state data

The following terms are defined in [RFC7950] and are used in this specification:

- o augment
- o data model
- o data node

The terminology for describing YANG data models is found in [RFC7950].

2.1. Tree Diagrams

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

3. Problem Statement

This document defines a YANG data model which allows user to configure the TACACS+ function on a network system. YANG model can be used with network management protocols such as NETCONF [RFC6241] to install, manipulate, and delete the configuration of network devices.

TACACS+ implementations in every device may vary greatly in terms of the data hierarchy and operations that they support. Therefore this draft proposes a model that can be augmented by standard extensions and vendor proprietary models.

4. Design of the Data Model

Although different vendors have different TACACS+ data model, there is a common understanding of what Terminal Access Controller Access Control System Plus (TACACS+) is. A network system usually has a TACACS+ functions which provides centralized validation of users attempting to gain access to a device or network access server.

TACACS+ services are maintained in a database on a TACACS server.

TACACS+ provides for separate and modular authentication, authorization, and accounting facilities and allows for a single

TACACS+ server to provide each service authentication, authorization, and accounting independently. Each service can be tied into its own database to take advantage of other services available on that server or on the network, depending on the capabilities of the server.

4.1. TACACS+ Modules Overview

The ietf-tacacs+ module augments the "/sys:system" path defined in the ietf-system module [RFC7317] with "tacacs" grouping defined in Section 3.2.

Under the 'tacacs' grouping, there are global-attributes container and a tacacs-templates container.

The global-attributes container is used to present the 'enable' and 'service-name' configuration and the global statistics information.

The tacacs-templates container is used to describe the tacacs configuration templates and operation templates.

Under tacacs-templates container, there are tacacs-servers container, ipv6-servers container, and host-servers container.

In the direction orthogonal to the tacacs container, presented are the commands. Those, in YANG terms, are the RPC commands. These RPC commands provide uniform APIs for resetting all statistics, resetting authentication statistics, resetting authorization statistics, resetting accounting statistics, and resetting common statistics.

The data model for tacacs has the following structure:

```

module: ietf-tacacs
  augment /sys:system:
    +--rw tacacs {tacacs}?
      +--rw global-attributes
      |   +--rw enable?                boolean
      |   +--ro total-templates?      uint32
      |   +--ro total-servers?        uint32
      |   +--rw service-name?         string
      +--rw tacacs-templates
        +--rw tacacs-template* [name]
          +--rw name                  string
          +--rw domain-include?       boolean
          +--rw timeout?              uint32
          +--rw quiet-time?           uint32
          +--rw shared-key?           password-extend
          +--rw source-ip?            inet:ipv4-address-no-zone
          +--rw domain-mode?          domain-include

```


+++ro pri-authen-srv?	inet:ipv4-address-no-zone
+++ro pri-common-srv?	inet:ipv4-address-no-zone
+++ro pri-author-srv?	inet:ipv4-address-no-zone
+++ro cur-authen-srv?	inet:ipv4-address-no-zone
+++ro cur-author-srv?	inet:ipv4-address-no-zone
+++ro sec-authen-srv-num?	uint32
+++ro sec-common-srv-num?	uint32
+++ro sec-author-srv-num?	uint32
+++ro pri-authen-port?	uint32
+++ro pri-common-port?	uint32
+++ro pri-author-port?	uint32
+++ro cur-authen-port?	uint32
+++ro cur-author-port?	uint32
+++ro authen-srv-connected-num?	uint32
+++ro authen-srv-disconnected-num?	uint32
+++ro authen-reqs-num?	uint32
+++ro authen-rsps-num?	uint32
+++ro authen-unknowns-num?	uint32
+++ro authen-timeouts-num?	uint32
+++ro authen-pkts-drop-num?	uint32
+++ro authen-passwords-change-num?	uint32
+++ro authen-logins-num?	uint32
+++ro authen-send-reqs-num?	uint32
+++ro authen-send-passwords-num?	uint32
+++ro authen-abort-reqs-num?	uint32
+++ro authen-connection-reqs-num?	uint32
+++ro authen-rsp-errs-num?	uint32
+++ro authen-rsp-fails-num?	uint32
+++ro authen-rsp-follows-num?	uint32
+++ro authen-get-data-num?	uint32
+++ro authen-get-password-num?	uint32
+++ro authen-get-user-num?	uint32
+++ro authen-rsps-pass-num?	uint32
+++ro authen-restart-num?	uint32
+++ro authen-no-process-num?	uint32
+++ro authen-time?	uint32
+++ro authen-errors-num?	uint32
+++ro author-srv-connected-num?	uint32
+++ro author-srv-disconnected-num?	uint32
+++ro author-reqs-num?	uint32
+++ro author-rsps-num?	uint32
+++ro author-unknowns-num?	uint32
+++ro author-timeouts-num?	uint32
+++ro author-pkts-drop-num?	uint32
+++ro author-reqs-exec-num?	uint32
+++ro author-ppp-num?	uint32
+++ro author-vpdn-num?	uint32
+++ro author-rsps-err-num?	uint32

```

    +--ro author-rsps-exec-num?          uint32
    +--ro author-rsps-ppp-num?           uint32
    +--ro author-rsps-vpdn-num?          uint32
    +--ro author-time?                   uint32
    +--ro author-reqs-not-process-num?    uint32
    +--ro author-errors-num?             uint32
    +--ro sec-accounting-servers-num?     uint32
    +--ro cur-account-port?              uint32
    +--ro pri-account-port?              uint32
    +--ro cur-account-srv?                inet:ipv4-address-no-zone
    +--ro pri-account-srv?                inet:ipv4-address-no-zone
    +--ro account-pkts-stop-num?          uint32
    +--ro account-rsps-pass-num?          uint32
    +--ro account-rsps-num?              uint32
    +--ro account-srvs-connected-num?     uint32
    +--ro account-pkts-rsps-num?          uint32
    +--ro account-reqs-num?              uint32
    +--ro account-srv-disconnected-num?   uint32
    +--ro account-rsps-errs-num?          uint32
    +--ro account-follow-rsps-num?        uint32
    +--ro account-reqs-not-process-num?    uint32
    +--rw tacacs-servers
      | +--rw tacacs-server* [server-ip server-type secondary-server net
work-instance public-net]
      |   +--rw server-ip                  inet:ipv4-address-no-zon
e
      |   +--rw server-type                server-type
      |   +--rw secondary-server            boolean
      |   +--rw network-instance            -> /ni:network-instances
/network-instance/name
      |   +--rw public-net                  boolean
      |   +--rw server-port?                uint32
      |   +--rw mux-mode-enable?            boolean
      |   +--ro server-current-state?       server-state
      |   +--ro current-srv?                boolean
      |   +--rw shared-key?                password-extend
      |   +--ro authen-srv-connected-num?    uint32
      |   +--ro authen-srv-disconnected-num? uint32
      |   +--ro authen-reqs-num?            uint32
      |   +--ro authen-rsps-num?            uint32
      |   +--ro author-srv-connected-num?    uint32
      |   +--ro author-srv-disconnected-num? uint32
      |   +--ro author-reqs-num?            uint32
      |   +--ro author-rsps-num?            uint32
      |   +--ro acct-reqs-num?              uint32
      |   +--ro acct-rsps-num?              uint32
      |   +--ro acct-srv-connected-num?      uint32
      |   +--ro acct-srv-disconnected-num?   uint32
    +--rw ipv6-servers
      | +--rw ipv6-server* [server-ip server-type secondary-server netwo
rk-instance]
      |   +--rw server-ip                  inet:ipv6-address-no-zon
e

```

```

    |      +---rw server-type                server-type
    |      +---rw secondary-server            boolean
    |      +---rw network-instance            -> /ni:network-instances
/network-instance/name
    |      +---rw server-port?                uint32
    |      +---rw mux-mode-enable?            boolean
    |      +---ro server-state?               server-state
    |      +---ro current-srv?                boolean
    |      +---rw shared-key?                 password-extend
    |      +---ro authen-srv-connected-num?   uint32
    |      +---ro authen-srv-disconnected-num? uint32
    |      +---ro authen-reqs-num?            uint32
    |      +---ro authen-rsps-num?            uint32
    |      +---ro author-srv-connected-num?   uint32
    |      +---ro author-srv-disconnected-num? uint32
    |      +---ro author-reqs-num?            uint32
    |      +---ro author-rsps-num?            uint32
    |      +---ro acct-reqs-num?              uint32
    |      +---ro acct-rsps-num?              uint32
    |      +---ro acct-srv-connected-num?     uint32
    |      +---ro acct-srv-disconnected-num?  uint32
    +---rw host-servers
        +---rw host-server* [server-host-name server-type secondary-serve
r network-instance public-net]
            +---rw server-host-name          string
            +---rw server-type                server-type
            +---rw secondary-server            boolean
            +---rw network-instance            -> /ni:network-instances
/network-instance/name
            +---rw public-net                 boolean
            +---rw server-port?                uint32
            +---rw mux-mode-enable?            boolean
            +---ro server-state?               server-state
            +---ro current-server?             boolean
            +---rw shared-key?                 password-extend
            +---ro authen-srv-connected-num?   uint32
            +---ro authen-srv-disconnected-num? uint32
            +---ro authen-reqs-num?            uint32
            +---ro authen-rsps-num?            uint32
            +---ro author-srv-connected-num?   uint32
            +---ro author-srv-disconnected-num? uint32
            +---ro author-reqs-num?            uint32
            +---ro author-rsps-num?            uint32
            +---ro acct-reqs-num?              uint32
            +---ro acct-rsps-num?              uint32
            +---ro acct-srv-connected-num?     uint32
            +---ro acct-srv-disconnected-num?  uint32

rpcs:
    +---x rest-all-statistics
    +---x reset-authen-statistics

```

```
----x reset-author-statistics
----x reset-account-statistics
----x reset-common-statistics
```

5. TACACS+ Module

```
<CODE BEGINS> file "ietf-tacacs@2018-06-25.yang"

module ietf-tacacs {
  namespace "urn:ietf:params:xml:ns:yang:ietf-tacacs";
  prefix tcs;

  import ietf-inet-types {
    prefix inet;
  }
  import ietf-network-instance {
    prefix ni;
  }
  import ietf-system {
    prefix sys;
  }

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";
  contact
    "WG Web:    <http://tools.ietf.org/wg/netmod/>
    WG List:    <mailto:netmod@ietf.org>

    Editor:     Guangying Zheng
                <mailto:zhengguangying@huawei.com>";
  description
    "This module provide defines a component that describe the
    configuration of TACACS+.";

  revision 2018-06-25 {
    description
      "Initial revision.";
    reference "foo";
  }

  typedef password-extend {
    type string {
      length "1..255";
    }
    description
      "now password extend is like string";
  }
}
```

```
typedef timezone-name {
  type string;
  description
    "A time zone name as used by the Time Zone Database,
    sometimes referred to as the 'Olson Database'.

    The exact set of valid values is an implementation-specific
    matter. Client discovery of the exact set of time zone names
    for a particular server is out of scope.";
  reference "RFC 6557: Procedures for Maintaining the Time Zone Database";
}
typedef server-state {
  type enumeration {
    enum "up" {
      description
        "The server is active.";
    }
    enum "down" {
      description
        "The server is inactive.";
    }
  }
  description
    "The type of tacacs server state";
}
typedef server-type {
  type enumeration {
    enum "authentication" {
      description
        "The server is an authentication server.";
    }
    enum "authorization" {
      description
        "The server is an authorization server.";
    }
    enum "accounting" {
      description
        "The server is an accounting server.";
    }
    enum "common" {
      description
        "The server is a common server.";
    }
  }
  description
    "The type of tacacs server";
}
typedef domain-include {
```

```
type enumeration {
  enum "no" {
    description
      "User name excludes domain.";
  }
  enum "yes" {
    description
      "User name includes domain.";
  }
  enum "original" {
    description
      "User name same as user input.";
  }
}
description
  "The type of domain mode";
}

feature tacacs {
  description
    "Indicates that the device can be configured as a tacacs
    client.";
}

grouping tacacs {
  container tacacs {
    if-feature tacacs;
    description
      "Container for TACACS configurations and operations.";
    container global-attributes {
      description
        "TACACS global attributes.";
      leaf enable {
        type boolean;
        default "false";
        description
          "Whether the TACACS server is enabled.";
      }
      leaf total-templates {
        type uint32;
        config false;
        description
          "Total number of TACACS templates configured.";
      }
      leaf total-servers {
        type uint32;
        config false;
      }
    }
  }
}
```

```

        description
            "Total number of TACACS servers configured.";
    }
    leaf service-name {
        type string {
            length "1..32";
        }
        description
            "TACACS service name.";
    }
}
container tacacs-templates {
    description
        "A set of TACACS templates.";
    list tacacs-template {
        key "name";
        description
            "List for tacacs template.";
        leaf name {
            type string;
            description
                "Name of a TACACS template, it is not case sensitive. The template n
ame can have alphabets a to z (case insensitive) and numbers from 0 to 9 or symb
ols ('.', '-', and '_').";
        }
        leaf domain-include {
            type boolean;
            default "true";
            description
                "Whether a domain name is included in a user name. By default, a use
r name contains the domain name.";
        }
        leaf timeout {
            type uint32 {
                range "1..300";
            }
            default "5";
            description
                "Server response timeout period. The default timeout period is 5 sec
onds.";
        }
        leaf quiet-time {
            type uint32 {
                range "1..255";
            }
            default "5";
            description
                "Time period after which the primary server restores to active. The
default time period is 5 minutes. The time period can be modified no matter whet
her users are using the TACACS template.";
        }
        leaf shared-key {
            type password-extend;
            description

```

"Shared key for a TACACS server. Configuring a shared key improves the communication security between a router and TACACS server. By default, no shared key is configured.";

```
    }
    leaf source-ip {
      type inet:ipv4-address-no-zone;
      description
        "Source IP address for a TACACS server.";
    }
    leaf domain-mode {
      type domain-include;
      default "yes";
      description
        "To configure domain Mode";
    }
    leaf pri-authen-srv {
      type inet:ipv4-address-no-zone;
      config false;
      description
        "IP address of the primary authentication server.";
    }
    leaf pri-common-srv {
      type inet:ipv4-address-no-zone;
      config false;
      description
        "IP address of the primary common server.";
    }
    leaf pri-author-srv {
      type inet:ipv4-address-no-zone;
      config false;
      description
        "IP address of the primary authorization server.";
    }
    leaf cur-authen-srv {
      type inet:ipv4-address-no-zone;
      config false;
      description
        "IP address of the authentication server being used.";
    }
    leaf cur-author-srv {
      type inet:ipv4-address-no-zone;
      config false;
      description
        "IP address of authorization server being used.";
    }
    leaf sec-authen-srv-num {
      type uint32;
      config false;
      description
        "Total number of configured secondary authentication servers in the
        template.";
```



```
    }
    leaf sec-common-srv-num {
        type uint32;
        config false;
        description
            "Total number of configured secondary common servers in the template
.";
    }
    leaf sec-author-srv-num {
        type uint32;
        config false;
        description
            "Total number of configured secondary authorization servers in the template.";
    }
    leaf pri-authen-port {
        type uint32;
        config false;
        description
            "Port of the primary authentication server.";
    }
    leaf pri-common-port {
        type uint32;
        config false;
        description
            "Port of the primary common server.";
    }
    leaf pri-author-port {
        type uint32;
        config false;
        description
            "Port of the primary authorization server.";
    }
    leaf cur-authen-port {
        type uint32;
        config false;
        description
            "Authentication server port being used.";
    }
    leaf cur-author-port {
        type uint32;
        config false;
        description
            "Authorization server port being used.";
    }
    leaf authen-srv-connected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client connected to the authentication server.";
```

```
    }
    leaf authen-srv-disconnected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client disconnected from the authentication server.";
    }
    leaf authen-reqs-num {
        type uint32;
        config false;
        description
            "Number of authentication requests. ";
    }
    leaf authen-rsps-num {
        type uint32;
        config false;
        description
            "Number of authentication responses.";
    }
    leaf authen-unknowns-num {
        type uint32;
        config false;
        description
            "Number of unknown authentication packets received by the TACACS client.";
    }
    leaf authen-timeouts-num {
        type uint32;
        config false;
        description
            "Number of times that authentication times out.";
    }
    leaf authen-pkts-drop-num {
        type uint32;
        config false;
        description
            "Number of times that authentication packets are dropped.";
    }
    leaf authen-passwords-change-num {
        type uint32;
        config false;
        description
            "Number of times that the password is changed for authentication.";
    }
    leaf authen-logins-num {
        type uint32;
        config false;
        description
            "Number of authentication logins.";
```

```
}
leaf authen-send-reqs-num {
  type uint32;
  config false;
  description
    "Number of authentication requests sent to server.";
}
leaf authen-send-passwords-num {
  type uint32;
  config false;
  description
    "Number of authentication password requests sent to the server.";
}
leaf authen-abort-reqs-num {
  type uint32;
  config false;
  description
    "Number of authentication abort requests sent to server.";
}
leaf authen-connection-reqs-num {
  type uint32;
  config false;
  description
    "Number of authentication connection requests sent to server.";
}
leaf authen-rsp-errs-num {
  type uint32;
  config false;
  description
    "Number of authentication error responses received from server.";
}
leaf authen-rsp-fails-num {
  type uint32;
  config false;
  description
    "Number of authentication response failures received from server.";
}
leaf authen-rsp-follows-num {
  type uint32;
  config false;
  description
    "Number of authentication Follow responses received from server.";
}
leaf authen-get-data-num {
  type uint32;
  config false;
  description
    "Number of authentication date responses received from server.";
```

```
    }
    leaf authen-get-password-num {
        type uint32;
        config false;
        description
            "Number of authentication password responses received from server.";
    }
    leaf authen-get-user-num {
        type uint32;
        config false;
        description
            "Number of authentication user responses received from server.";
    }
    leaf authen-rsps-pass-num {
        type uint32;
        config false;
        description
            "Number of authentication-pass responses received from server.";
    }
    leaf authen-restart-num {
        type uint32;
        config false;
        description
            "Number of authentication-restart responses received from server.";
    }
    leaf authen-no-process-num {
        type uint32;
        config false;
        description
            "Number of authentication requests that are not processed.";
    }
    leaf authen-time {
        type uint32;
        config false;
        description
            "Time (in tick) taken to complete the authentication.";
    }
    leaf authen-errors-num {
        type uint32;
        config false;
        description
            "Number of authentication errors.";
    }
    leaf author-srv-connected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client connected to the authorizati
on server.";
```

```
    }
    leaf author-srv-disconnected-num{
        type uint32;
        config false;
        description
            "Number of times that the TACACS client disconnected from the author
ization server.";
    }
    leaf author-reqs-num {
        type uint32;
        config false;
        description
            "Number of authorization requests. ";
    }
    leaf author-rsps-num {
        type uint32;
        config false;
        description
            "Number of authorization responses.";
    }
    leaf author-unknowns-num {
        type uint32;
        config false;
        description
            "Number of unknown authorization packets received by TACACS client."
;
    }
    leaf author-timeouts-num {
        type uint32;
        config false;
        description
            "Number of times that authorization times out.";
    }
    leaf author-pkts-drop-num {
        type uint32;
        config false;
        description
            "Number of times that authorization packets are dropped.";
    }
    leaf author-reqs-exec-num {
        type uint32;
        config false;
        description
            "Number of authorization requests for execute.";
    }
    leaf author-ppp-num {
        type uint32;
        config false;
        description
            "Number of authorization requests for PPP.";
```

```
    }
    leaf author-vpdn-num{
        type uint32;
        config false;
        description
            "Number of authorization requests for VPDN.";
    }
    leaf author-rsps-err-num {
        type uint32;
        config false;
        description
            "Number of authorization error responses.";
    }
    leaf author-rsps-exec-num {
        type uint32;
        config false;
        description
            "Number of authorization execute responses.";
    }
    leaf author-rsps-ppp-num {
        type uint32;
        config false;
        description
            "Number of authorization PPP responses.";
    }
    leaf author-rsps-vpdn-num {
        type uint32;
        config false;
        description
            "Number of authorization VPDN responses.";
    }
    leaf author-time {
        type uint32;
        config false;
        description
            "Time (in tick) taken to complete authorization.";
    }
    leaf author-reqs-not-process-num {
        type uint32;
        config false;
        description
            "Number of authorization requests that are not processed.";
    }
    leaf author-errors-num {
        type uint32;
        config false;
        description
            "Number of authorization errors.";
```

```
}
leaf sec-accounting-servers-num {
  type uint32;
  config false;
  description
    "Number of secondary accounting servers in the template.";
}
leaf cur-account-port {
  type uint32;
  config false;
  description
    "Accounting server port being used.";
}
leaf pri-account-port {
  type uint32;
  config false;
  description
    "Port of the primary accounting server.";
}
leaf cur-account-srv {
  type inet:ipv4-address-no-zone;
  config false;
  description
    "Accounting server port being used.";
}
leaf pri-account-srv {
  type inet:ipv4-address-no-zone;
  config false;
  description
    "Primary accounting server.";
}
leaf account-pkts-stop-num {
  type uint32;
  config false;
  description
    "Number of responses to accounting-stop packets.";
}
leaf account-rsps-pass-num {
  type uint32;
  config false;
  description
    "Number of responses to accounting-pass packets.";
}
leaf account-rsps-num {
  type uint32;
  config false;
  description
    "Number of responses to accounting requests.";
```

```
    }
    leaf account-srvs-connected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client connected to the accounting
server.";
    }
    leaf account-pkts-rsps-num {
        type uint32;
        config false;
        description
            "Number of responses to accounting-start packets.";
    }
    leaf account-reqs-num {
        type uint32;
        config false;
        description
            "Number of accounting requests sent to the server.";
    }
    leaf account-srv-disconnected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client disconnected from the account
ting server.";
    }
    leaf account-rsps-errs-num {
        type uint32;
        config false;
        description
            "Number of abnormal accounting responses received from the server.";
    }
    leaf account-follow-rsps-num {
        type uint32;
        config false;
        description
            "Number of accounting Follow responses received from server.";
    }
    leaf account-reqs-not-process-num {
        type uint32;
        config false;
        description
            "Number of accounting requests that are not processed.";
    }
    container tacacs-servers {
        description
            "A set of TACACS servers.";
        list tacacs-server {
            key "server-ip server-type secondary-server network-instance public-
net";
```



```
description
  "TACACS IPV4 server. A maximum 32 servers can be configured in one
template ";

      leaf server-ip {
        type inet:ipv4-address-no-zone;
        description
          "Server IPv4 address. Must be a valid unicast IP address.";
      }
    leaf server-type {
      type server-type;
      description
        "Server type: authentication/authorization/accounting/common.";
    }
    leaf secondary-server {
      type boolean;
      description
        "Whether the server is secondary. By default, a server is a seco
ndary server.";
    }
    leaf network-instance {
      type leafref {
        path "/ni:network-instances/ni:network-instance/ni:name";
      }
      description
        "VPN instance name.";
    }
    leaf public-net {
      type boolean;
      description
        "Set the public-net.";
    }
    leaf server-port {
      type uint32 {
        range "1..65535";
      }
      default "49";
      description
        "Server port. Value range: 1-65535. The default port number is 4
9.";
    }
    leaf mux-mode-enable {
      type boolean;
      default "false";
      description
        "Whether the MUX mode is enabled for the server. By default, the
MUX mode is disabled.";
    }
    leaf server-current-state {
      type server-state;
      config false;
      description
```

```
        "Server running status.";
    }
    leaf current-srv {
        type boolean;
        default "false";
        config false;
        description
            "Whether the server is being used.";
    }
    leaf shared-key {
        type password-extend;
        description
            "Shared key for a TACACS server. Configuring a shared key improves the communication security between a router and TACACS server. By default, no shared key is configured.";
    }
    leaf authen-srv-connected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client connected to the authentication server.";
    }
    leaf authen-srv-disconnected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client disconnected from the authentication server.";
    }
    leaf authen-reqs-num {
        type uint32;
        config false;
        description
            "Number of authentication requests. ";
    }
    leaf authen-rsps-num {
        type uint32;
        config false;
        description
            "Number of authentication responses.";
    }
    leaf author-srv-connected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client connected to the authorization server.";
    }
    leaf author-srv-disconnected-num {
        type uint32;
        config false;
        description
```

```
        "Number of times that the TACACS client disconnected from the au
thorization server.";
    }
    leaf author-reqs-num {
        type uint32;
        config false;
        description
            "Number of authorization requests. ";
    }
    leaf author-rsps-num {
        type uint32;
        config false;
        description
            "Number of authorization responses.";
    }
    leaf acct-reqs-num {
        type uint32;
        config false;
        description
            "Number of accounting requests. ";
    }
    leaf acct-rsps-num {
        type uint32;
        config false;
        description
            "Number of accounting responses.";
    }
    leaf acct-srv-connected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client connected to the account
ing server.";
    }
    leaf acct-srv-disconnected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client disconnected from the ac
counting server.";
    }
}
container ipv6-servers {
    description
        "A set of TACACS servers.";
    list ipv6-server {
        key "server-ip server-type secondary-server network-instance";
        description
            "TACACS IPV6 server. A maximum 32 servers can be configured in one
template ";
        leaf server-ip {
```

```
    type inet:ipv6-address-no-zone;
    description
        "Server IPv6 address. Must be a valid unicast IP address.";
}
leaf server-type {
    type server-type;
    description
        "Server type: authentication/authorization/accounting/common.";
}
leaf secondary-server {
    type boolean;
    description
        "Whether the server is secondary. By default, a server is a secondary server.";
}
leaf network-instance {
    type leafref {
        path "/ni:network-instances/ni:network-instance/ni:name";
    }
    description
        "Configure the vpn-instance name.";
}
leaf server-port {
    type uint32 {
        range "1..65535";
    }
    default "49";
    description
        "Server port. Value range: 1-65535. The default port number is 49.";
}
leaf mux-mode-enable {
    type boolean;
    default "false";
    description
        "Whether the MUX mode is enabled for the server. By default, the MUX mode is disabled.";
}
leaf server-state {
    type server-state;
    config false;
    description
        "Server running status.";
}
leaf current-srv {
    type boolean;
    default "false";
    config false;
    description
        "Whether the server is being used.";
}
```

```
    leaf shared-key {
        type password-extend;
        description
            "Shared key for a TACACS server. Configuring a shared key improves the communication security between a router and TACACS server. By default, no shared key is configured.";
    }
    leaf authen-srv-connected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client connected to the authentication server.";
    }
    leaf authen-srv-disconnected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client disconnected from the authentication server.";
    }
    leaf authen-reqs-num {
        type uint32;
        config false;
        description
            "Number of authentication requests. ";
    }
    leaf authen-rsps-num {
        type uint32;
        config false;
        description
            "Number of authentication responses.";
    }
    leaf author-srv-connected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client connected to the authorization server.";
    }
    leaf author-srv-disconnected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client disconnected from the authorization server.";
    }
    leaf author-reqs-num {
        type uint32;
        config false;
        description
            "Number of authorization requests. ";
    }
    leaf author-rsps-num {
```

```

        type uint32;
        config false;
        description
            "Number of authorization responses.";
    }
    leaf acct-reqs-num {
        type uint32;
        config false;
        description
            "Number of accounting requests. ";
    }
    leaf acct-rsps-num {
        type uint32;
        config false;
        description
            "Number of accounting responses.";
    }
    leaf acct-srv-connected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client connected to the account
ing server.";
    }
    leaf acct-srv-disconnected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client disconnected from the ac
counting server.";
    }
}
}
container host-servers {
    description
        "A set of TACACS host servers.";
    list host-server {
        key "server-host-name server-type secondary-server network-instance
public-net";
        description
            "TACACS host server. A maximum 32 servers can be configured in one
template.";
        leaf server-host-name {
            type string {
                length "1..255";
            }
            description
                "Host name of TACACS server. Host name, Can include character '
', '-', '_' and lowercase or uppercase letters and digit, at least include one l
etter or digit.";
        }
        leaf server-type {
            type server-type;
            description

```

```
        "Server type: authentication/authorization/accounting/common.";
    }
    leaf secondary-server {
        type boolean;
        description
            "Whether the server is secondary. By default, a server is a secondary server.";
    }
    leaf network-instance {
        type leafref {
            path "/ni:network-instances/ni:network-instance/ni:name";
        }
        description
            "VPN instance name.";
    }
    leaf public-net {
        type boolean;
        description
            "Set the public-net.";
    }
    leaf server-port {
        type uint32 {
            range "1..65535";
        }
        default "49";
        description
            "Server port. Value range: 1-65535. The default port number is 49.";
    }
    leaf mux-mode-enable {
        type boolean;
        default "false";
        description
            "Whether the MUX mode is enabled for the server. By default, the MUX mode is disabled.";
    }
    leaf server-state {
        type server-state;
        config false;
        description
            "Server running status.";
    }
    leaf current-server {
        type boolean;
        default "false";
        config false;
        description
            "Whether the server is being used.";
    }
    leaf shared-key {
        type password-extend;
```

```
        description
            "Shared key for a TACACS server. Configuring a shared key improv
es the communication security between a router and TACACS server. By default, no
shared key is configured.";
    }
    leaf authen-srv-connected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client connected to the authent
ication server.";
    }
    leaf authen-srv-disconnected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client disconnected from the au
thentication server.";
    }
    leaf authen-reqs-num {
        type uint32;
        config false;
        description
            "Number of authentication requests. ";
    }
    leaf authen-rsps-num {
        type uint32;
        config false;
        description
            "Number of authentication responses.";
    }
    leaf author-srv-connected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client connected to the authori
zation server.";
    }
    leaf author-srv-disconnected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client disconnected from the au
thorization server.";
    }
    leaf author-reqs-num {
        type uint32;
        config false;
        description
            "Number of authorization requests. ";
    }
    leaf author-rsps-num {
        type uint32;
        config false;
```



```

        description
            "Number of authorization responses.";
    }
    leaf acct-reqs-num {
        type uint32;
        config false;
        description
            "Number of accounting requests. ";
    }
    leaf acct-rsps-num {
        type uint32;
        config false;
        description
            "Number of accounting responses.";
    }
    leaf acct-srv-connected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client connected to the account
ing server.";
    }
    leaf acct-srv-disconnected-num {
        type uint32;
        config false;
        description
            "Number of times that the TACACS client disconnected from the ac
counting server.";
    }
}
}
}
}
}
description
    "Grouping for tacacs";
}

augment "/sys:system" {
    uses tacacs;
    description
        "Augment the system module";
}

rpc rest-all-statistics {
    description
        "Reset All Statistics.";
}
rpc reset-authen-statistics {
    description

```

```
    "Reset authentication statistics of the TACACS server.";
  }
  rpc reset-author-statistics {
    description
      "Reset authorization statistics of the TACACS server.";
  }
  rpc reset-account-statistics {
    description
      "Reset accounting statistics of the TACACS server.";
  }
  rpc reset-common-statistics {
    description
      "Reset common statistics of the TACACS server.";
  }
}
```

<CODE ENDS>

6. Security Considerations

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

7. 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-tacacs
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 [RFC7950].

Name: ietf-tacacs
Namespace: urn:ietf:params:xml:ns:yang: ietf-tacacs
Prefix: tcs
Reference: RFC XXXX

8. Normative References

- [RFC1492] Finseth, C., "An Access Control Protocol, Sometimes Called TACACS", RFC 1492, DOI 10.17487/RFC1492, July 1993, <<https://www.rfc-editor.org/info/rfc1492>>.
- [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>>.
- [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>>.
- [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>>.

- [RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <<https://www.rfc-editor.org/info/rfc7317>>.
- [RFC792] Postel, J., "Internet Control Message Protocol", RFC 792, September 1981.
- [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>>.
- [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>>.

Authors' Addresses

Guangying Zheng
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: zhengguangying@huawei.com

Michael Wang
Huawei Technologies, Co., Ltd
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: wangzitao@huawei.com

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

Email: lana.wubo@huawei.com