NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 23, 2021

Q. Wu
Huawei
I. Bryskin
Individual
H. Birkholz
Fraunhofer SIT
X. Liu
Volta Networks
B. Claise
Cisco

February 19, 2021

A YANG Data model for ECA Policy Management
draft-ietf-netmod-eca-policy-01

Abstract

This document defines a YANG data model for Event Condition Action (ECA) policy management. The ECA policy YANG module provides the ability to delegate some network management functions to the server (e.g., a NETCONF or RESTCONF server) which can take simple and instant action when a trigger condition on the managed objects is met.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 23, 2021.

Table of Contents

1. Introduction

Traditional approaches for the network to automatically perform corrective actions in response to network events have been largely built on centralized policy-based management [RFC3198]. With centralized network management, the managed object state or operational state spanning across the devices needs to be retrieved by the client from various servers. However, there are issues associated with centralized network management:

- Centralized network management incurs massive data collection and processing, the resource consumption (e.g., network bandwidth usage, the state to be maintained) is huge;
- Centralized network management leads to slow reaction to the network changes when large amounts of managed object state from devices needs to be collected and correlated at the central point where decisions about resource adjustment are made;
- Centralized network management cannot control or influence management behavior within the server if the server is not connected to any network or the existing configuration on the server has major errors;
- Centralized network management doesn't scale well when thousands of devices need to send hundreds of event notifications, or millions of managed data objects needs to be polled by the client;

A more effective complementary approach to centralized network management is to delegate some of network management functions (e.g.,log dump task routine) to servers in the network and allow servers to self monitor state changes of managed objects. Accordingly, there is a need for a service in the server to provide continuous performance monitoring, detect defects and failures, and take corrective action.

This document defines an ECA Policy management YANG data model. The ECA Policy YANG allows the client to move some of network management tasks to the server (e.g., a NETCONF or RESTCONF server), which provides the ability to control the configurations and monitor state parameters, and take simple and instant action on the server when a trigger condition on the system state is met.

The data model in this document is designed to be compliant with the Network Management Datastore Architecture (NMDA) [RFC8342].

2. Conventions used in this document

2.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying [RFC2119] significance.

The following terms are defined in [RFC3198][RFC6241][RFC7950] and are not redefined here:

- Policy Decision Point (PDP)
- Policy Enforcement Point (PEP)
- Provisioned Policy
- Server
- Client
- Event

This document uses the following terms:

Condition: Condition can be seen as a logical test on local managed object that, if satisfied or evaluated to be true, causes the action to be carried out.

Action: Update or invocation on local managed object attributes.

ECA Event: The input to the ECA logic that initiates the processing derived from an extensible list of platform event types.

Server Event: An event that happens in the server for which a Notification could be generated in an Event Stream subscription.

Datastore Event: An event that happens within a datastore within the server for which a Notification could be generated in a datastore subscription.

Timer Event: A pseudo-event in the server that allows ECA logic to be invoked periodically.

Diagnostic Event: A pseudo-event initiated by the client to test ECA logic.

Self Monitoring: Automatic monitoring of resources to ensure the optimal functioning with respect to the defined requirements.

Self Healing: Automatic discovery and correction of faults; automatically applying all necessary Actions to bring the system back to normal operation.

Policy Variable (PV): Represents datastore states that change (or "vary"), and that is set or evaluated by software.

PV-Source: Represents an XPath result, which contains one of four data types: Boolean, Number, String, and Node Set.

PV-Result: Represents the value of the result of an Policy Variable evaluation.

2.2. Tree Diagrams

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

3. Overview of ECA YANG Data Model

A ECA policy rule is read as: when event occurs in a situation where condition is true, then action is executed. Therefore, ECA comprises three key elements: event, associated conditions, and actions. These three elements should be pushed down and configured on the server by the client. If the action is rejected by the server during ECA policy execution, the action should be rolled back and cleaned up.

3.1. ECA Policy Variable and Value

ECA policy variable (PV) generically represents datastore states that change (or "vary"), and that is set or evaluated by software. The value of ECA policy variable is used for modeling values and constants used in policy conditions and actions. In policy, conditions and actions can abstract information as "policy variables" to be evaluated in logical expressions, or set by actions, e.g., the policy condition has the semantics "variable matches value" while policy action has the semantics "set variable to value".

In ECA, two type of policy variables are defined, pv-source variable and pv-result variable. pv-source variable represents an XPath

expression input, which contains one of four data types: Boolean, Number, String, and Node Set while pv-result variable represents the value of the result of an Policy Variable evaluation.

- A pv-source is always config = true.
- A pv-result is always config = false.
- A single anydata cannot be used for all values since it is only allowed to contain child nodes. Separate scalar and nodeset values are needed.

Each ECA policy variable has the following two attributes:

- Name with Globally unique or ECA unique scope ;
- Type either pv-source or pv-result;

The following operations are allowed with/on a PV:

- initialize (with a constant/enum/identity);
- set (with contents of another same type PV);
- read (retrieve datastore contents pointed by the specified same type XPath/sub-tree);
- write (modify configuration data in the datastore with the PV's content/value);
- function calls or RPC in a form of F(arg1,arg2,...), where F is an identity of a function from extendable function library, arg1,arg2,etc are PVs respectively, the function's input parameters, with the result returned in result policy variable.

PVs could also be a source of information sent to the client in notification messages.

PVs could be also used in condition expressions.

The model structure for the Policy Variable is shown below:

```
        +--rw policy-variables
        |  +--rw policy-variable* [name]
        |     +--rw name                              string
        |     +--rw type                              identityref
        |     +--rw (xpath-value-choice)?
        |        +--:(policy-source)
        |        |  +--rw (pv-source)
        |        |     +--:(xpath-expr)
        |        |     |  +--rw xpath-expr?          yang:xpath1.0
        |        |     +--:(scalar-constant)
        |        |     |  +--rw scalar-constant?     string
        |        |     +--:(nodeset-constant)
        |        |        +--rw nodeset-constant?    <anydata>
        |        +--:(policy-result)
        |           +--rw (pv-result)
        |              +--:(scalar-value)
        |              |  +--rw scalar-value?        string
        |              +--:(nodeset-value)
        |                 +--rw nodeset-value?       <anydata>
```

3.2. ECA Event

The ECA Event is any subscribable event notification either explicitly defined in a YANG module (e.g., interface management model) supported by the server or a event stream conveyed to the server via YANG Push subscription. The ECA event are used to keep track of state of changes associated with one of multiple operational state data objects in the network device.

Each ECA Event can be classified into server event, datastore event, timer event, diagnostics event and has the following common attributes:

- event-name, the name of ECA event;
- event-type, typical examples of ECA event type include server event, datastore event, timer event and diagnostic event.

For server event, the following additional attributes are defined:

- event-stream, typical example of event stream is NETCONF stream.
- event-module, the name of YANG module associated with the ECA event.
- event, it is event stream conveyed to the server.

For datastore event, the following additional attributes are defined:

datastore, the name of the datastore, typical example of datastore is running, operational state datastores.

data-path, in the form of XPATH expression.

data, it is event notification defined in a YANG module.

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

- Register the event recording its name and using the referred PUSH subscription trigger as definition of the event firing trigger;
- Auto-configure the event's ECA input in the form of local PVs using the PUSH subscription's filters;
- 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.

The model structure for the ECA Event is shown below:

```
     +--rw events
     |  +--rw event* [event-name]
     |     +--rw event-name               string
     |     +--rw event-type?              identityref
     |     +--rw policy-variable*         -> /gncd/policy-variables/policy-variab
le/name
     |     +--rw local-policy-variable*   -> /gncd/ecas/eca/policy-variable/name
     |     +--rw (type-choice)?
     |        +--:(server-event)
     |        |  +--rw event-stream?      string
     |        |  +--rw event-module?      string
     |        |  +--rw event?             <anydata>
     |        +--:(datastore-event)
     |        |  +--rw datatore?          string
     |        |  +--rw data-path?         string
     |        |  +--rw data?              <anydata>
     |        +--:(timer-event)
     |        +--:(diagnostics-event)
```

3.3. ECA Condition

The ECA Condition is the logical expression that is specified in a form of Xpath expression and evaluated to TRUE or FALSE. The XPath expression specifies an arbitrary logical/mathematical expression; The elements of the ECA Condition expression are referred by the XPaths pointing to referred datastore states.

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

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 ECA Action is executed.

The model structure for the condition is shown below:

```
     +--rw conditions
     |  +--rw condition* [name]
     |     +--rw name                     string
     |     +--rw (expression-choice)?
     |        +--:(xpath)
     |           +--rw condition-xpath?   string
```

3.3.1. Mapping Policy Variables to XPath Variables

Policy variables are mapped to XPath variable bindings so they can be referenced in the XPath expression for a Condition.

- The 'name' leaf value for the policy variable is mapped to the local-name of the XPath variable. No namespace is used for ECA variables. E.g., the policy variable named 'foo' would be accessible with a variable reference '$foo'.

- The local-name 'USER' is reserved and defined in NACM. The server SHOULD provide the USER variable as NACM is implemented.

- XPath variables can be used in 2 main ways in an expression:

1) anchor of a path-expr

```
$node-set-variable/child1/nested2
```

2) right-hand side of a primary-expr

```
/foo[name = $scalar-variable]
```

- It cannot be used in the middle of a path-expr

```
/interfaces/$node-set-variable/child1/nested2     // NOT OK
```

- Since a variable is a primary expression it can be used in XPath expression constructions anywhere a primary-expr is allowed

```
$nodeset-variable1 | $ nodeset-variable2

($min-length + $avg-length) < $last-length
```

- The values of all available policy variables are updated by the server (if required) before the XPath expression is evaluated. The variable binding value MUST NOT change while the XPath expression is being evaluated. If multiple references to the same variable exist in an XPath expression, they MUST resolve to the same value in each instance.

Example: "/test1[name=$badfan] and /test2[name=$badfan]"
The same value of 'badfan' is expected in each instance.

- If a variable reference cannot be resolved because no policy variable with that name is accessible to the ECA under evaluation, then an eca-exception notification SHOULD be generated, and the XPath evaluation MUST be terminated with an error.

3.3.2. ECA XPath Context

All XPath expressions used in ECA share the following XPath context definition.

- The set of namespace declarations is the set of all modules loaded into the server now. Prefix bindings can reference the set of namespace URIs for this set of modules.

- All names SHOULD be namespace-qualified. There is no default namespace to use if no namespace is specified. If no namespace is used then the XPath step matches the local-name in all namespaces.

- The function library is the core function library defined in [XPATH], the functions defined in Section 10 of [RFC7950], and the ECALIB functions defined in this document Section 3.5.1.

- The set of variable bindings is set to all policy variables that are visible to the ECA under evaluation. This includes the local-policy-variable and policy-variable entries configured for the 'eca' entry. Since pv-source values can reference other policy variables, the order that these fields are set is significant.

- The accessible tree is all state data in the server, and the running configuration datastore. The root node has all top-level data nodes in all modules as children.

- The context node for all ECA XPath evaluation is the root node.

3.3.3. ECA Evaluation Exceptions

Not all errors can be detected at configuration time. Error that occur while ECA logis is being evaluated will cause the server to generate an eca-exception notification.

If the ECA is scheduled one time, an exception to ECA entry execution will be generated if the error occurs. If the ECA is scheduled periodically and duplicated exception notification is generated in the second period interval, ECA entry execution will be disabled automatically and in addition ECA entry disable exception will be generated and sent to the local client.

```
identity eca-exception-reason {
  description
    "Base of all values for the 'reason' leaf in the
    eca-exception notification.";
}

identity varbind-unknown {
  base eca-exception-reason;
  description
    "The requested policy variable binding is not defined.
     The variable binding cannot be resolved in the XPath
     evaluation.";
}
identity func-invoke-error {
  base eca-exception-reason;
  description
    "The function call is invoked and return false output.";
 }
identity rpc-call-error {
  base eca-exception-reason;
  description
    "The rpc call is invoked and return false output.";
 }
identity eca-entry-disable {
  base eca-exception-reason;
  description
    "The ECA entry is disabled if the same exception occurs more than once
     in the periodical ECA.";
 }
// Additional exceptions can be added as needed
 notification eca-exception {
   description
     "This notification is sent when some error occurs
      while the server is processing ECA logic.";
    leaf reason {
      type eca-exception-reason;
    }
 }
```

3.4. ECA Action

The ECA Action list consists of updates or invocations on local managed object attributes and a set of actions are defined as follows, which will be performed when the corresponding event is triggered:

- sending one time notification

- (re-)configuration scheduling - scheduling one time or periodic (re-)configuration in the future
- stopping current ECA;
- invoking the same ECA recursively;

Three points are worth noting:

- 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 ECA model requires each notification to carry mandatory information, such as event and (event scope unique) notification names.
- Multiple ECA 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 actions is shown below:

```
        +--rw actions
        |  +--rw time-schedule!
        |  |  +--rw period?   centiseconds
        |  +--rw action* [name]
        |     +--rw name                    string
        |     +--rw action-element* [name]
        |     |  +--rw name                       string
        |     |  +--rw action-type?               identityref
        |     |  +--rw (action-operation)?
        |     |     +--:(action)
        |     |     |  +--rw next-period          boolean
        |     |     |  +--rw action-name?
        |     |     |          -> /gnca/actions/action/name
        |     |     +--:(function-call)
        |     |     |  +--rw function-call
        |     |     |     +--rw func-name       leafref
        |     |     |     +--rw policy-source   leafref
        |     |     |     +--rw policy-result   leafref
        |     |     |  +--:(rpc-operation)
        |     |     |  |  +--rw rpc-operation
        |     |     |  |     +--rw rpc-name?          string
        |     |     |  |     +--rw nc-action-xpath?   string
```

3.5. ECA

An ECA container includes:

- ECA name.

- List of local PVs and global PVs. As mentioned, These PVs could be configured as dynamic (their instances appear/disappear with start/stop of the ECA execution) or as static (their instances exist as long as the ECA is configured). Global PV will be shared by multiple ECA instances while local PVs are within the scope of a specific ECA instance.

- Normal CONDITION-ACTION list: configured conditions each with associated actions to be executed if the condition is evaluated to TRUE

Note that this document currently focuses on one event with multiple conditions and actions case. How different ECAs do not impact each other if they share PVs and other components is not in the scope of this document at this moment.

3.5.1. ECA XPath Function Library (ECALIB)

A set of common event PVs need to be set for every invocation of condition or action logic:

```
$event-type      (string)
$event-name    (string)
```

For event-type = "server-event"

```
$event-stream   (string)
$event-module   (string)
$event-name     (string)
$event          (node-set)
```

The condition can use these PVs directly in an expression
An expression can access client-configured PVs of course

```
$event/child[name=$some-global-var] > 10
```

For event-type = "datastore"

```
$datastore  (string)
$data-path  (string)
$data       (node-set)
```

The data is defined to be a container with the requested data as child nodes

```
$data/interface[type=$gigabit-eth] // (node-set is an array of data nodes, usua
lly
siblings)
```

A standard func call should be defined to specify operation on policy variables and xpath expression and store func result.

```
//Increment count by one each time increment-func is invoked
boolean function increment-func(number count)

//Decrement count by one each time decrement-func is invoked
boolean function decrement-func(number count)

//Exit the loop to monitor specific event
boolean function exit-func()

//Continue the loop to monitor the specific event
boolean function continue-func()

//set iteration variable as true if count variable is equal to or greater than 1
//set iteration variable as false if count variable is zero
boolean function match-func (string expr,number count,boolean iteration)
```

```
// check every 5 seconds until the same event occurs 2 times
  sustained-event("$event/child[type=$some-global-var]/descendant[$leaf1 > 10]",
5, 2)

boolean function sustained-event (string expr, number interval, number count)
        test expression 'expr' once per 'interval'. Keep testing once per
        interval until true result reached, i.e., both xpath expression is
        evaluated to true and 'count' number of interval on specific data
        object has been tested true
        (e.g., the same event occurs 'count' times )Return true if condition
        tested true for count intervals; Returns false otherwise;

// check the event record every 5 seconds and filter the event record with
   constraint of a specific descendant node to the event record root node
   filtered-event("$event/child/descendant[$leaf1 > 10]", "$event",5)

boolean function filtered-event (string input-expr,string output-expr,number
        interval)test expression 'expr'once per 'interval' and generate event
        record output represented by 'output-expr' based on 'input-expr'.
        Note than 'output-expr'and 'input-expr'share the same root node;

A standard rpc should be defined to specify the operation on the event stream
// suppress the event stream corresponding to XPATH expression
boolean rpc event-duplication-suppress(string expr)
```

The ECA XPath function library is expected to grow over time and additional standard or vendor function libraries should be possible. The server should provide a read-only list of ECA function libraries supported. How it is exposed to the client is beyond scope of this document.

```
        +--rw eca-func-libs
           +--rw eca-function* [func-name]
           |   +--rw func-name    string
           +--rw eca-rpc* [rpc-name]
           |  +--rw rpc-name     string
           +--rw eca-name    -> /gncd/ecas/eca/name
```

Note that ECA accesses specific datastores in the same way as YANG Push [RFC8641]. The difference is condition expression is introduced to further filter nodes in the node set and the policy variable is introduced to keep the intermediate states during the interaction between the local client and the server.

4. ECA YANG Model (Tree Structure)

The following tree diagrams [RFC8340] provide an overview of the data model for the "ietf-eca" module.

```
module: ietf-eca
  +--rw gncd
     +--rw policy-variables
     |  +--rw policy-variable* [name]
     |     +--rw name                              string
     |     +--rw type                              identityref
     |     +--rw (xpath-value-choice)?
     |        +--:(policy-source)
     |        |  +--rw (pv-source)
     |        |     +--:(xpath-expr)
     |        |     |  +--rw xpath-expr?          yang:xpath1.0
     |        |     +--:(scalar-constant)
     |        |     |  +--rw scalar-constant?     string
     |        |     +--:(nodeset-constant)
     |        |        +--rw nodeset-constant?    <anydata>
     |        +--:(policy-result)
     |           +--rw (pv-result)
     |              +--:(scalar-value)
     |              |  +--rw scalar-value?        string
     |              +--:(nodeset-value)
     |                 +--rw nodeset-value?       <anydata>
     |
     +--rw events
     |  +--rw event* [event-name]
     |     +--rw event-name               string
     |     +--rw event-type?              identityref
     |     +--rw policy-variable*         -> /gncd/policy-variables/policy-variab
le/name
     |     +--rw local-policy-variable*   -> /gncd/ecas/eca/policy-variable/name
     |     +--rw (type-choice)?
     |        +--:(server-event)
     |        |  +--rw event-stream?      string
     |        |  +--rw event-module?      string
     |        |  +--rw event?             <anydata>
     |        +--:(datastore-event)
     |        |  +--rw datatore?          string
     |        |  +--rw data-path?         string
     |        |  +--rw data?              <anydata>
     |        +--:(timer-event)
     |        |  +--rw start-time      yang:date-and-time
     |        |  +--rw duration        centiseconds
     |        |  +--rw repeat-option   identityref
     |        |  +--rw repeat-time-len centiseconds
     |        +--:(diagnostics-event)
     +--rw conditions
     |  +--rw condition* [name]
     |     +--rw name                     string
     |     +--rw (expression-choice)?
     |        +--:(xpath)
     |           +--rw condition-xpath?   string
```

```
     +--rw actions
     |  +--rw time-schedule!
     |  |  +--rw period?   centiseconds
     |  +--rw action* [name]
     |     +--rw name              string
     |     +--rw action-element* [name]
     |     |  +--rw name                       string
     |     |  +--rw action-type?               identityref
     |     |  +--rw (action-operation)?
     |     |  |  +--:(action)
     |     |  |  |  +--rw next-period         boolean
     |     |  |  |  +--rw action-name?
     |     |  |  |          -> /gnca/actions/action/name
     |     |  |  +--:(function-call)
     |     |  |  |  +--rw function-call
     |     |  |  |     +--rw func-name       leafref
     |     |  |  |     +--rw policy-source   leafref
     |     |  |  |     +--rw policy-result   leafref
     |     |  |  +--:(rpc-operation)
     |     |  |  |  +--rw rpc-operation
     |     |  |  |     +--rw rpc-name?          string
     |     |  |  |     +--rw nc-action-xpath?   string
     |  +--rw ecas
     |  |  +--rw eca* [name]
     |  |     +--rw name                string
     |  |     +--rw username            string
     |  |     +--rw event-name          string
     |  |     +--rw policy-variable* [name]
     |  |     |  +--rw name                            leafref
     |  |     |  +--rw is-static?                      boolean
     |  |     +--rw condition-action* [name]
     |  |     |  +--rw name         string
     |  |     |  +--rw condition*   -> /gncd/conditions/condition/name
     |  |     |  +--rw action?      -> /gncd/actions/action/name
     |  |     +---x start
     |  |     +---x stop
     |  |     +---x next-action
     |  +--rw eca-func-libs
     |     +--rw eca-function* [func-name]
     |     |  +--rw func-name    string
     |     +--rw eca-rpc* [rpc-name]
     |     |  +--rw rpc-name     string
     |     +--rw eca-name    -> /gncd/ecas/eca/name

  notifications:
    +---n eca-exception
    |  +--ro reason?   identityref
    +---n custom-notification
```

```
       +--ro eventTime       yang:date-and-time
       +--ro event-type?     identityref
       +--ro (type-choice)?
       |   +--:(server-event)
       |   |  +--ro event-stream?      string
       |   |  +--ro event-module?      string
       |   |  +--ro policy-result      leafref
       |   +--:(datastore-event)
       |   |  +--ro datatore?          string
       |   |  +--ro data-path?         string
       |   |  +--ro policy-result      leafref
```

5. ECA YANG Module

<CODE BEGINS> file "ietf-eca@2019-10-28.yang"

```
module ietf-eca {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-eca";
  prefix gnca;

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-netconf-acm {
   prefix nacm;
   reference
    "RFC8341: Network Configuration Access Control Model";
  }
  organization
    "IETF Network Configuration (NETCONF) Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>
     Editor:   Qin Wu
               <mailto:bill.wu@huawei.com>
     Editor:   Igor Bryskin
               <mailto:Igor.Bryskin@huawei.com>
     Editor:   Henk Birkholz
               <mailto:henk.birkholz@sit.fraunhofer.de>
     Editor:   Xufeng Liu
               <mailto:xufeng.liu.ietf@gmail.com>
     Editor:   Benoit Claise
               <mailto:bclaise@cisco.com>
     Editor:   Andy Bierman
               <mailto:andy@yumaworks.com>
     Editor:   Alexander Clemm
               <mailto:ludwig@clemm.org>";
```

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

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

  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, sustained-event.";
  }

  identity sustained-event {
    description
      "Identity for standard sustained-event function call,
       the input variables for sustained-event include string
       expr, number interval, number count. Keep testing
       expression 'expr'once per interval until false result
       reached.Return true if condition tested true
       for count intervals; Returns false otherwise.";
  }

  identity plus {
    description
      "Identity for standard plus function call, the input
       variables for plus function call include src policy argument
```

```
      and dst policy arugment.";
  }

  identity minus {
    description
      "Identity for standard minus function call, the input
       variables for plus function call include src policy argument
       and dst policy arugment.";
  }

  identity multiply {
    description
      "Identity for standard multiply function call, the input
       variables for multiply function call include src policy argument
       and dst policy arugment.";
  }

  identity divide {
    description
      "Identity for standard divide function call, the input
       variables for multiply function call include src policy argument
       and dst policy arugment.";
  }

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

  identity event-type {
    description
      "Base identity for Event Type.";
  }

  identity server-event {
    base event-type;
    description
      "Identity for server event.";
  }

  identity datastore-event {
    base event-type;
    description
      "Identity for datastore event.";
  }

  identity timer-event {
```

```
    base event-type;
    description
      "Identity for timer event.";
  }

  identity diagnostics-event {
    base event-type;
    description
      "Identity for diagnostics event.";
  }

  identity eca-exception-reason {
     description
       "Base of all values for the 'reason' leaf in the
       eca-exception notification.";
   }

   identity varbind-unknown {
     base eca-exception-reason;
     description
       "The requested policy variable binding is not defined.
        The variable binding cannot be resolved in the XPath
        evaluation.";
   }

  typedef centiseconds {
    type uint32;
    description
      "A period of time, measured in units of 0.01 seconds.";
  }

  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 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.";
  }

      grouping scalar-value {
      leaf scalar-value {
        type string;
        description
          "Represents an XPath simple value that has an
           XPath type of Boolean, String, or Number.
           This value will be converted to an XPath type,
           as needed.

           A YANG value is encoded as a string using the same
           rules as the 'default' value for the data type.

           An eca-exception notification is generated if a scalar
           XPath value is used in a path expression, where a
           node-set is expected. Normally XPath will treat this result
           as an empty node-set, but this is an ECA programming error.";
      }
    }

    grouping nodeset-value {
      anydata nodeset-value {
         description
           "Represents an XPath node set. A 'node-set' anydata node
            with no child data nodes represents an empty node-set.
            Each child node in within this anydata structure
            represents a subtree that is present in the XPath
            node-set.

            An XPath node-set is not required to contain a top-level
            YANG data node.  It is not required to contain an entire
            complete subtree.

            It is am implementation-specific manner how a
            representation of YANG 'anydata' nodes are mapped
```

```
            to specific YANG module schema definitions.";
      }
    }
      grouping scalar-constant {
      leaf scalar-constant {
        type string;
        description
          "Represents an XPath simple value that has an
           XPath type of Boolean, String, or Number.
           This value will be converted to an XPath type,
           as needed.

           A YANG value is encoded as a string using the same
           rules as the 'default' value for the data type.

           An eca-exception notification is generated if a scalar
           XPath value is used in a path expression, where a
           node-set is expected. Normally XPath will treat this result
           as an empty node-set, but this is an ECA programming error.";
      }
    }

    grouping nodeset-constant {
      anydata nodeset-constant {
         description
           "Represents an XPath node set. A 'node-set' anydata node
            with no child data nodes represents an empty node-set.
            Each child node in within this anydata structure
            represents a subtree that is present in the XPath
            node-set.

            An XPath node-set is not required to contain a top-level
            YANG data node.  It is not required to contain an entire
            complete subtree.

            It is am implementation-specific manner how a
            representation of YANG 'anydata' nodes are mapped
            to specific YANG module schema definitions.";
      }
    }
  grouping pv-source {
    choice pv-source {
      mandatory true;
      description
        "A PV source represents an XPath result, which contains
         one of four data types: Boolean, Number, String,
         and Node Set.  XPath defines mechanisms to covert
         values between these four types.
```

```
          The 'xpath-expr' leaf is used to assign the PV source
          to the result of an arbitrary XPath expression.
          The result of this expression evaluation is used
          internally as needed. The result may be any one of
          the XPath data types.

          The 'scalar-constant' leaf is used to represent a Boolean,
          String, or Number XPath constant value.

          The 'nodeset-constant' anydata structure is used to
          represent a constant XPath node-set.";

        leaf xpath-expr {
          type yang:xpath1.0;
          description
            "Contains an XPath expression that must be evaluated
             to produce an XPath value. [section X.X] describes
             the XPath execution environment used to process this
             object.";
        }

        case scalar-constant {
          uses scalar-constant;
        }
        case nodeset-constant {
          uses nodeset-constant;
        }
      }
    }

    grouping pv-result {
      choice pv-result {
        mandatory true;
        description
          "Represents the value of the result of an
           Policy Variable evaluation.

           The 'scalar-value' leaf is used to represent a Boolean,
           String, or Number XPath result value.

           The 'nodeset-value' anydata structure is used to represent
           an XPath node-set result.";

        case scalar-value {
          uses scalar-value;
        }
        case nodeset-value {
          uses nodeset-value;
```

```
      }
    }
  }

  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 xpath-value-choice {
      description
        "The type of a policy variable may be either a common
         primative type like boolean or a type from existing
         schema node referenced by an XPath string.";
      /*case scalar {
        uses scalar-value;
      }
      case nodeset {
        uses nodeset-value;
      }*/
    case policy-source {
    uses pv-source;
   }
   case policy-result {
     uses pv-result;
   }
    }
  }

  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 next-period {
             type boolean;
             description
               "invoke the same eca recursively if the next period
               is set to true.";
           }
             leaf action-name {
              type leafref {
                path "/gncd/actions/action/name";
              }
              description
                "The operation is to execute a configured ECA Action.";
            }
          } // action
     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-name {
                type string;
                description
                  "The name of function call to be called";
               }
        leaf policy-source {
         type leafref {
          path "/gncd/policy-variables/policy-variable/name";
       }
      description
       "The policy source.";
     }
     leaf policy-result {
     type leafref {
       path "/gncd/policy-variables/policy-variable/name";
      }
      description
       "The policy result.";
     }
            }
          } // 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 rpc-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.";
            }
          }
        } // 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.";
            }
          }
        }
      }*/
    }
  }

  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 period {
        type centiseconds;
        description
          "Duration of time that should occur between periodic
           push updates, in units of 0.01 seconds.";
      }
    }
  }

  container gncd {
    nacm:default-deny-all;
    description
      "Top level container for Generalized Network Control Automation
       (gncd).";
    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;
      }
    }
    container events {
      description
        "Container of ECA events.";
      list event {
        key "event-name";
        description
          "A list of events used as the triggers of ECAs.";
        leaf event-name {
          type string;
          description
            "The name of the event.";
        }
        leaf event-type {
          type identityref {
            base event-type;
          }
          description
            "The type of the event.";
        }
        leaf-list policy-variable {
          type leafref {
```

```
            path "/gncd/policy-variables/"
              + "policy-variable/name";
             }
          description
           "global policy variables, which
            are shared by all ECA scripts.";
           }
        leaf-list local-policy-variable {
          type leafref {
            path "/gncd/ecas/eca/policy-variable/name";
          }
          description
            "local policy variables, which
             are 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 server event and datastore event.";
          case server-event {
            leaf event-stream {
              type string;
              description
                "The name of a subscribed stream .";
            }
            leaf event-module {
              type string;
              description
                "The name of YANG data module associated with the subscribed
                 stream.";
            }
            anydata event {
              description
                "This anydata value MUST Contain the absolute XPath
                 expression identifying the element path to the node that is
                 associated with subscribed stream.";
            }
          }
          case datastore-event {
            leaf datatore {
              type string;
              description
                "The name of a datatore from which applications
                 subscribe to updates.";
            }
            leaf data-path {
              type string;
```

```
              description
                "The absolute XPath expression identifying the
                 element path to the node that is associated with
                 subscribed stream..";
            }
            anydata data {
              description
                "This anydata value MUST Contain the node that is
                            associated with the data path.";
            }
          }
          case timer-event {
               leaf start-time {
                  type yang:date-and-time;
                  description
                   "This object specifies the scheduled start date/time to trigge
r
                    timer event.";
               }
               leaf duration {
                 type centiseconds;
                 description
                   "This object specifies duration of the timer event execution.";
               }
              leaf repeat-option {
                 type centiseconds;
                 description
                   "This object indicate repeat option, e.g., repeat everyday, eve
ryweek,
                    everymoth,everyyear or every specfiied time length.";
               }
              leaf repeat-len {
                 type centiseconds;
                 description
                   "This object specifies the time length in 0.01 seconds after wh
ich
                    the timer event is executed for the duration.";
               }
           }
          case diagnostics-event;
        }
      }
    }
    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.";
          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.";
            }
          }
        }
      }
    }
    container actions {
      description
        "Container of ECA Actions.";
        uses time-schedule-container {
          description
            "Specifying the time schedule to execute this ECA
             Action.
             If not specified, the ECA Action is executed one time immediately
             when it is called.";
        }
      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;
        }
      }
    }
    container ecas {
      description
        "Container of ECAs.";
      list eca {
        key "name";
        description
          "A list of ECAs";
        leaf name {
          type string;
          description
            "A string name to uniquely identify an ECA globally.";
        }
     leaf username {
      type string;
         mandatory true;
         description
           "Name of the user for the session.";
       }
        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.";
          leaf name {
          type leafref {
            path "/gncd/policy-variables/policy-variable/name";
          }
        }
           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";
          ordered-by user;
        description
          "A list of Condition-Actions, which are configured
           conditions each with associated actions to be executed
           if the condition is evaluated to TRUE. The server can do
           multiple action when the condition is true. If the next-period
           is set to true, condition-action will be executed recursively.
           It is also possbile to require multiple conditions to be true
           in order to do one action.";
        leaf name {
          type string;
          description
            "A string name uniquely identify a Condition-Action
             within this ECA.";
        }
        leaf-list condition {
          type leafref {
            path "/gncd/conditions/condition/name";
          }
          description
            "The reference to a configured condition.";
        }
        leaf action {
          type leafref {
            path "/gncd/actions/action/name";
          }
          description
            "The reference to a configured action.";
        }
      }
      action start {
        description
          "Start to execute this ECA. The start action is invoked
           by the local client when the event type is set to diagnostic
          event.";
      }
      action stop {
        description
          "Stop the execution of this ECA. The stop action is invoked
           by the local client when the event type is set to diagnostic
           event.";
      }
```

```
        action next-action {
          description
            "Resume the execution of this ECA to complete the next
             action. The next action is invoked by the local client
             when the event type is set to diagnostic event.";
        }
      }
    }
 container eca-func-libs {
  description
    "Container of ECA Function Libraries.";
  list eca-function {
    key func-name;
    description
     "A list of ECA standard function.";
    leaf func-name {
     type string;
    description
    "A string name to uniquely identify an ECA standard function.";
   }
}
  list rpc-function {
    key rpc-name;
    description
     "A list of ECA standard function.";
    leaf rpc-name {
   type string;
   description
     "A string name to uniquely identify an ECA standard RPC.";
    }
  }
   leaf eca-name {
    type leafref {
     path "/gncd/ecas/eca/name";
    }
    description
     "The reference to a configured ECA.";
    }
   } // eca-scripts
  }

    notification eca-exception {
      description
        "This notification is sent when some error occurs
         while the server is processing ECA logic.";
       leaf reason {
         type identityref {
         base eca-exception-reason;
```

```
        }
       }
    }
 notification custom-notification {
      description
        "This notification is sent when some error occurs
         while the server is processing ECA logic.";
       leaf eventTime {
         type yang:date-and-time;
          description
           "The event occurrence time";
       }
      leaf event-type {
          type identityref {
            base event-type;
          }
          description
            "The type of the event.";
      }
       choice type-choice {
          description
            "The type of an event, including server event and datastore event.";
          case server-event {
            leaf event-stream {
              type string;
              description
                "The name of a subscribed stream .";
            }
            leaf event-module {
              type string;
              description
                "The name of YANG data module associated with the subscribed
                 stream.";
            }
            anydata event {
              description
                "This anydata value MUST Contain the absolute XPath
                 expression identifying the element path to the node that is
                 associated with subscribed stream.";
            }
          }
          case datastore-event {
            leaf datatore {
              type string;
              description
                "The name of a datatore from which applications
                 subscribe to updates.";
            }
```

```
            leaf data-path {
              type string;
              description
                "The absolute XPath expression identifying the
                 element path to the node that is associated with
                 subscribed stream..";
            }
            anydata data {
              description
                "This anydata value MUST Contain the node that is
                 associated with the data path.";
            }
          }
        }
    }
}
```

<CODE ENDS>

6. Security Considerations

The YANG modules defined in this document MAY be accessed via the RESTCONF protocol [RFC8040] or NETCONF protocol ([RFC6241]). The lowest RESTCONF or NETCONF layer requires that the transport-layer protocol provides both data integrity and confidentiality, see Section 2 in [RFC8040] and [RFC6241]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH)[RFC6242] . The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

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

- /gnca:gncd/gnca:policy-variables/gnca:policy-variable/gnca:name

- /gnca:gncd/gnca:events/gnca:event/gnca:name

- /gnca:gncd/gnca:conditions/gnca:condition/gnca:name
- /gnca:gncd/gnca:actions/gnca:action/gnca:name
- /gnca:gncd/gnca:ecas/gnca:eca/gnca:name
- /gnca:gncd/gnca:ecas/gnca:eca/gnca:username
- /gnca:gncd/gnca:eca-func-libs/gnca:eca-function/gnca:func-name

7. IANA Considerations

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested to be made:

```
---------------------------------------------------------------------
   URI: urn:ietf:params:xml:ns:yang:ietf-eca
   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-eca
   Namespace:    urn:ietf:params:xml:ns:yang:ietf-eca
   Prefix:       gnca
   Reference:    RFC xxxx
---------------------------------------------------------------------
```

8. Acknowledges

Igor Bryskin, Xufeng Liu, Alexander Clemm, Henk Birkholz, Tianran Zhou contributed to an earlier version of [GNCA]. We would like to thank the authors of that document on event response behaviors delegation for material that assisted in thinking that helped improve this document. We also would like to thanks Tom Petch,Juergen Schoenwaelder,Randy Preshun,Lingli Deng, Chang Liu, Yunbo Yan,Jonathan Hansford, Daniel King, Dhruv Dhody, Michale Wang, Xiaopeng Qin, Yu Yang, Haoyu Song, Tianran Zhou,Aihua Guo,Nicola Sambo,Giuseppe Fioccola for valuable review on this document.

9. Contributors

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Alex Clemm
Futurewei
Email: ludwig@clemm.org

Qiufang Ma
Huawei
Email: maqiufang1@huawei.com

Chongfeng Xie
China Telecom
Email: xiechf@ctbri.com.cn

Diego R. Lopez
Telefonica
Email:diego.r.lopez@telefonica.com

10. References

10.1. Normative References

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

[RFC3460] Moore, B., Ed., "Policy Core Information Model (PCIM) Extensions", RFC 3460, DOI 10.17487/RFC3460, January 2003, <https://www.rfc-editor.org/info/rfc3460>.

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

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

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

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

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

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

10.2. Informative References

[RFC3198] Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J., and S. Waldbusser, "Terminology for Policy-Based Management", RFC 3198, DOI 10.17487/RFC3198, November 2001, <https://www.rfc-editor.org/info/rfc3198>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

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

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

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

Appendix A. ECA Condition Expression Examples

Here are two examples of Condition Expression:

(a) a condition that only includes data store states and constants, for example:

TE metric of Link L in Topology T greater than 100,
it can be expressed as follows:

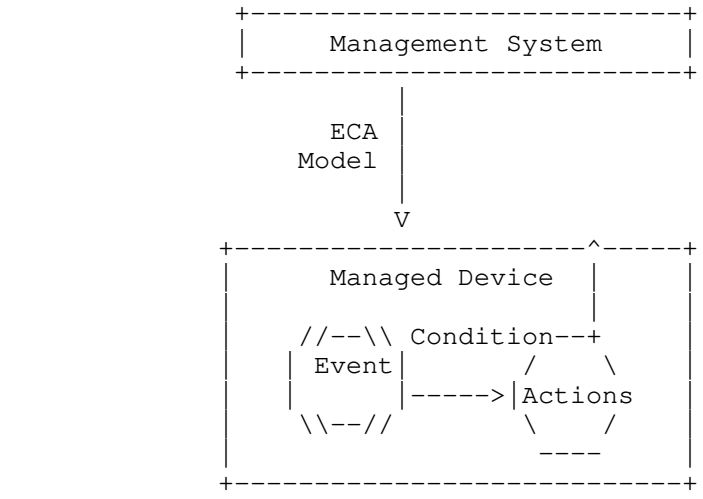
"/nw:networks/nw:network[network-id='T']/nt:link[link-id='L']/tet:te\
/tet:te-link-attributes/tet:te-delay-metric > 100"

(b) a condition that also includes a Policy Variable, for example:

Allocated bandwidth of Link L in Topology T greater than 75% of
what is stored in Policy Variable B, it can be expressed as follows:

"/nw:networks/nw:network[network-id='T']/nt:link[link-id='L']/tet:te\
/tet:te-link-attributes/tet:max-resv-link-bandwidth\
> (ietf-eca:policy-variables/policy-variable[name='B']/value) * 0.75"

Appendix B. Usage Example of Smart Filter using Server Event Trigger



The management system designs a new ECA policy based on monitored objects in ietf-interfaces module that support threshold checking and pushes down the ECA policy to control interface behavior in the managed device that supports NETCONF/RESTCONF protocol operation, i.e.,scan all interfaces for a certain type every 5 seconds and check the counters or status, return an array of interface entries (XPath node-set) that match the search and suppress reporting of duplicated events if all conditions are evaluated into true. The XML example snippet is shown as below:

```
<gnca>
    <policy-variables>
        <policy-variable>
```

```
            <name>event-repeat-count</name>
            <scalar-constant>0</scalar-constant>
        </policy-variable>
        <policy-variable>
            <name>interface-statistics-event</name>
            <xpath-expr>if:interfaces/if:interface[if:type=if:gigabitEthernet,
     if:oper-status=down]</xpath-expr>
        </policy-variable>
    </policy-variables>
    <events>
        <event>
            <event-name>interface-self-monitoring</event-name>
            <event-type>server-event</event-type>
            <event-stream>NETCONF</event-stream>
            <event-module>ietf-interfaces</event-module>
            <event>if:interfaces/if:interface[if:type=if:gigabitEthernet]</event>
        </event>
    </events>
    <conditions>
        <condition>
            <name>if-monitoring-condition1</name>
            <condition-xpath>event[if:oper-status=down]</condition-xpath>
        </condition>
        <condition>
            <name>if-monitoring-condition2</name>
            <condition-xpath>event[if:oper-status!=down]</condition-xpath>
        </condition>
        <condition>
            <name>if-monitoring-condition3</name>
            <condition-xpath>event-repeat-count >1 </condition-xpath>
        </condition>
        <condition>
            <name>if-monitoring-condition4</name>
            <condition-xpath>event-repeat-count <=1 </condition-xpath>
        </condition>
    </conditions>
    <actions>
        <time-schedule>
            <period>5</period>
        </time-schedule>
        <action>
            <name>if-matched-statistics1</name>
            <action-element>
                <name>event-filter-action</name>
                <func-name>filtered-event</func-name>
                <policy-source>interface-statistics-event</policy-source>
                <policy-result>event</policy-result>
            </action-element>
```

```
            <action-element>
                <name>increment-action</name>
                <func-name>increment-function</func-name>
                <policy-source>event-repeat-count</policy-source>
                <policy-result>event-repeat-count</policy-result>
            </action-element>
            <action-element>
                <name>suppress-action</name>
                <rpc-operation>
                    <name>suppress-notification</name>
                </rpc-operation>
            </action-element>
            <action-element>
                <name>continue-check-action</name>
                <func-name>match-function</func-name>
                <policy-source>interface-statistics-event</policy-source>
                <policy-source>event-repeat-count</policy-source>
                <policy-result>next-period</policy-result>
            </action-element>
        </action>
        <action>
            <name>if-matched-statistics2</name>
            <action-element>
                <name>event-filter-action</name>
                <func-name>filtered-event</func-name>
                <policy-source>interface-statistics-event</policy-source>
                <policy-result>event</policy-result>
            </action-element>
            <action-element>
                <name>increment-action</name>
                <func-name>increment-function</func-name>
                <policy-source>event-repeat-count</policy-source>
                <policy-result>event-repeat-count</policy-result>
            </action-element>
            <action-element>
                <name>continue-check-action</name>
                <func-name>match-function</func-name>
                <policy-source>interface-statistics-event</policy-source>
                <policy-source>event-repeat-count</policy-source>
                <policy-result>next-period</policy-result>
            </action-element>
        </action>
        <action>
            <name>if-matched-statistics3</name>
            <action-element>
                <name>decrement-action</name>
                <func-name>decrement-function</func-name>
                <policy-source>event-repeat-count</policy-source>
```

```
                <policy-result>event-repeat-count</policy-result>
            </action-element>
            <action-element>
                <name>exit-action</name>
                <func-name>exit-func</func-name>
            </action-element>
        </action>
    </actions>
    <ecas>
        <eca>
            <name>interface-eca-handling</name>
            <user-name>Bob</user-name>
            <event-name>interface-self-monitoring</event-name>
            <condition-action>
                <name>smart-filter1</name>
                <condition>if-monitoring-condition1</condition>
                <condition>if-monitoring-condition3</condition>
                <action>
                    <name>if-matched-statistics1</name>
                    <action-element>
                        <name>event-filter-action</name>
                    </action-element>
                    <action-element>
                        <name>increment-action</name>
                    </action-element>
                    <action-element>
                        <name>suppress-action</name>
                    </action-element>
                    <action-element>
                        <name>continue-check-action</name>
                    </action-element>
                </action>
            </condition-action>
            <condition-action>
                <name>smart-filter2</name>
                <condition>if-monitoring-condition1</condition>
                <condition>if-monitoring-condition4</condition>
                <action>
                    <name>if-matched-statistics2</name>
                    <action-element>
                        <name>event-filter-action</name>
                    </action-element>
                    <action-element>
                        <name>increment-action</name>
                    </action-element>
                    <action-element>
                        <name>continue-check-action</name>
                    </action-element>
```

```
                </action>
            </condition-action>
            <condition-action>
                <name>smart-filter3</name>
                <condition>if-monitoring-condition2</condition>
                <action>
                    <name>if-matched-statistics3</name>
                    <action-element>
                        <name>decrement-action</name>
                    </action-element>
                    <action-element>
                        <name>exit-action</name>
                    </action-element>
                </action>
            </condition-action>
        </eca>
    </ecas>
    <eca-func-libs>
        <eca-function>
            <func-name>filtered-event</func-name>
        </eca-function>
        <eca-function>
            <func-name>increment-function</func-name>
        </eca-function>
        <eca-function>
            <func-name>decrement-function</func-name>
        </eca-function>
        <eca-function>
            <func-name>exit-function</func-name>
        </eca-function>
        <eca-function>
            <func-name>match-function</func-name>
        </eca-function>
        <eca-rpc>
            <rpc-name>event-duplication-suppress</rpc-name>
        </eca-rpc>
        <eca-name>interface-eca-handling</eca-name>
    </eca-func-libs>
</gnca>

// This custom-notification is only sent when there is no duplicated event to occ
ur.
<custom-notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-11-21T13:51:00Z</eventTime>
  <event-type>server-event</event-type>
  <event-stream>NETCONF</event-stream>
  <event-module>ietf-interfaces</event-module>
  <event>if:interfaces/if:interface[if:type=if:gigabitEthernet]</event>
          <interfaces
```

```
            xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
            xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
          <interface>
            <name>GE0</name>
            <type>ianaift:gigabitEthernet</type>
            <enabled>false</enabled>
          </interface>
           .....
          <interface>
            <name>GE1</name>
            <type>ianaift:gigabitEthernet</type>
            <enabled>true</enabled>
            ...
          </interface>
          .....
          <interface>
            <name>GE2</name>
            <type>ianaift:gigabitEthernet</type>
            ...
            <enabled>true</enabled>
          </interface>
  </eca-report>
</custom-notification>
```

In this example, the event name is set to 'interface-self-monitoring', the event type is set to 'server-event', the function name of ECA function libraries is set to 'sustained-event', 'increment-function','decrement-function','match-function','exit-function' the rpc name of ECA function libraries is set to 'event-duplication-suppress',the name of 'condition-action' is corresponding to standard function calls described above. The pseudo code of ECA logic can be described as follows:

```
count = 0;
while { next-period = true}
if(interface is down ) {
      event= filtered event;//eca exception will be notified to the local client
if invoking filtered event fails
      count++;
      if(count > 1){
       suppress event;//eca exception will be notified to the local client if inv
oking filtered event fails
       next-period = true;
       exit;
     }else if( count <= 1) {
      next-period = true;
      call custom-notification;
      continue;
     }
   }else if ( interface is not down){
     next-period = false;
     count=0;
     exit;
  }
}
```

Appendix C. Usage Example of Router Log Dump using Timer Event Trigger

Use a watchdog to dump the router log every 180 seconds to a flash. The XML example snippet is shown as below:

```
<gnca>
    <policy-variables>
        <policy-variable>
            <name>syslog-remote-info</name>
            <xpath-expr>syslog:syslog/syslog:actions/syslog:remote</xpath-expr>
        </policy-variable>
    </policy-variables>
    <events>
        <event>
            <event-name>log-dump-monitoring</event-name>
            <start-time>2020-10-21T13:51:00Z</start-time>
            <duration>12000</duration>
            <repeat-option>everyminutes<repeat-option>
            <repeat-time-length>3<repeat-time-length>
        </event>
    </events>
    <actions>
        <action>
            <name>log-dump-statistics</name>
            <action-element>
                <name>log-dump-action</name>
                <rpc-name>syslog-remote-output</rpc-name>
```

```
                <nc-action-xpath>syslog-remote-info</nc-action-xpath>
            </action-element>
        </action>
    </actions>
    <ecas>
        <eca>
            <name>log-dump-handling</name>
            <user-name>Bob</user-name>
            <event-name>log-dump-monitoring</event-name>
            <condition-action>
                <name>cron-log-monitoring</name>
                <action>
                    <name>log-dump-statistics</name>
                    <action-element>
                        <name>syslog-remote-output</name>
                    </action-element>
                </action>
            </condition-action>
        </eca>
    </ecas>
    <eca-func-libs>
        <eca-rpc>
            <rpc-name>syslog-remote-output</rpc-name>
        </eca-rpc>
        <eca-name>log-dump-handling</eca-name>
    </eca-func-libs>
</gnca>
```

Appendix D. Usage Example of High CPU Utilization Troubleshooting

It is usually found that at times the CPU utilization spikes up for a very short period of time and at indeterminate times. ECA to be executed in the network device can be used to detect CPU utilization,e.g.,It is triggered when the CPU utilization goes above 60% and also output stack, cpu, fan statistics information to a flash. The XML example snippet is shown as below:

```
<gnca>
    <policy-variables>
        <policy-variable>
            <name>stack-info</name>
            <xpath-expr>hw:hardware/hw:components/hw:component[hw:class=stack]</x
path-expr>
        </policy-variable>
        <policy-variable>
            <name>fan-info</name>
            <xpath-expr>hw:hardware/hw:components/hw:component[hw:class=fan]</xpa
th-expr>
        </policy-variable>
        <policy-variable>
```

```
            <name>sensor-info</name>
            <xpath-expr>hw:hardware/hw:components/hw:component[hw:class=sensor]</
xpath-expr>
        </policy-variable>
    </policy-variables>
    <events>
        <event>
            <event-name>cpu-util-monitoring</event-name>
            <event-type>server-event</event-type>
            <event-stream>NETCONF</event-stream>
            <event-module>ietf-hardware</event-module>
            <event>hw:hardware/hw:components/hw:component[hw:class=cpu]</event>
        </event>
    </events>
    <conditions>
        <condition>
            <name>cpu-utilization-condition</name>
            <condition-xpath>event/sensor-data[value>60,value-type=percentile]</c
ondition-xpath>
        </condition>
    </conditions>
    <actions>
        <action>
            <name>cpu-info-filter</name>
            <action-element>
                <name>cpu-info-dump-action1</name>
                <func-name>filtered-event</func-name>
                <policy-source>event/sensor-data[value>60,value-type=percentile]<
/policy-source>
                <policy-result>stack-info</policy-result>
            </action-element>
            <action-element>
                <name>cpu-info-dump-action2</name>
                <func-name>filtered-event</func-name>
                <policy-source>event/sensor-data[value>60,value-type=percentile]<
/policy-source>
                <policy-result>fan-info</policy-result>
            </action-element>
            <action-element>
                <name>cpu-info-dump-action3</name>
                <func-name>filtered-event</func-name>
                <policy-source>event/sensor-data[value>60,value-type=percentile]<
/policy-source>
                <policy-result>sensor-info</policy-result>
            </action-element>
        </action>
        <action>
            <name>cpu-info-output</name>
            <action-element>
                <name>cpu-info-dump-action1</name>
                <rpc-name>cpu-log-dump</rpc-name>
                <nc-action-xpath>stack-info</nc-action-xpath>
            </action-element>
```

```
            <action-element>
                <name>cpu-info-dump-action2</name>
                <rpc-name>cpu-log-dump</rpc-name>
                <nc-action-xpath>fan-info</nc-action-xpath>
            </action-element>
            <action-element>
                <name>cpu-info-dump-action3</name>
                <rpc-name>cpu-log-dump</rpc-name>
                <nc-action-xpath>sensor-info</nc-action-xpath>
            </action-element>
            <action-element>
                <name>cpu-info-dump-action4</name>
                <rpc-name>cpu-log-dump</rpc-name>
                <nc-action-xpath>event/sensor-data[value>60,value-type=percentile
]</nc-action-xpath>
            </action-element>
        </action>
    </actions>
    <ecas>
        <eca>
            <name>cpu-util-handling</name>
            <user-name>Bob</user-name>
            <event-name>cpu-util-monitoring</event-name>
            <condition-action>
                <name>cpu-log-monitoring</name>
                <condition>cpu-utilization-condition</condition>
                <action>
                    <name>cpu-info-filter</name>
                    <action-element>
                        <name>cpu-info-dump-action1</name>
                    </action-element>
                    <action-element>
                        <name>cpu-info-dump-action2</name>
                    </action-element>
                    <action-element>
                        <name>cpu-info-dump-action3</name>
                    </action-element>
                </action>
            </condition-action>
            <condition-action>
                <name>cpu-log-printing</name>
                <action>
                    <name>cpu-info-output</name>
                    <action-element>
                        <name>cpu-info-dump-action1</name>
                    </action-element>
                    <action-element>
                        <name>cpu-info-dump-action2</name>
                    </action-element>
```

```
                    <action-element>
                        <name>cpu-info-dump-action3</name>
                    </action-element>
                    <action-element>
                        <name>cpu-info-dump-action4</name>
                    </action-element>
                </action>
            </condition-action>
        </eca>
    </ecas>
    <eca-func-libs>
        <eca-function>
            <func-name>filtered-event</func-name>
        </eca-function>
        <eca-rpc>
            <rpc-name>cpu-log-dump</rpc-name>
        </eca-rpc>
        <eca-name>cpu-util-handling</eca-name>
    </eca-func-libs>
</gnca>
```

Appendix E. Open Issues tracking

- Relationship with I2NSF YANG capability-data-model.
- What is the Abstraction level to express policies and intent?
- Where are policies executed?
- When to detect and resolve policy conflicts?
- Who is interested in interoperable policy representations / languages?

Appendix F. Changes between Revisions

v00 -v01

- Clarify the relationship between centralized network management and network function delegation;
- Add clarification text on the ECA definition;
- Other Editorial changes;

v09 - v10

- Rewrite ECA Model Self Monitoring Usage Example;

- Add usage Example of High CPU Utilization Troubleshooting;
- Add usage Example of Router Log Dump using Timer Event Trigger;
- Reintroduce iterate action, function call and rpc call action type. These action types are exchanged between local client and the server.
- Move notification operation as separate notification since the notification is exchange between the management system and the server.

v08 - v09

- Add ECA function libraries list in the ECA model.
- Subtree and data node path fixing in the security section.

v07 - v08

Replace ECA model usage example with self monitoring usage example in the appendix.

Clean up references.

Add a new section to discuss Mapping Policy Variables to XPath Variables.

Add a new section to discuss ECA XPath Context.

Add a new section to discuss ECA Evaluation Exceptions.

Rewrite Introduction to highlight elevator pitch.

Replace implicit variable and explicit variable with pv-source variable and pv-result variable.

Take out function-call, cleanup-condition-action list, execution list, policy argument container, eca-script list at this moment.

v06 - v07

- Reuse alarm notification event received on an event stream (RFC 8639) in ECA logic;
- Represent ECA condition expression only in the form of Xpath expression;

- Add ECA condition expression example in the appendix;
- Add ECA model usage example in the appendix;
- Remove the section to discuss the relation with YANG push;
- Remove the dependency to SUPA framework draft;
- Remove smart filter extension example in the Appendix.
- Bind ECA script with condition expression in the model.

v05 - v06

- Decouple ECA model from NETCONF protocol and make it applicable to other network mangement protocols.
- Move objective section to the last section with additional generic objectives.

v04 - v05

- Harmonize with draft-bryskin and add additional attributes in the models (e.g., policy variable, func call enhancement, rpc execution);
- ECA conditions part harmonization;
- ECA Event, Condition, Action, Policy Variable and Value definition;
- Change ietf-event.yang into ietf-eca.yang and remove ietf-event-trigger.yang

v02 - v03

- Usage Example Update: add an usage example to introduce how to reuse the ietf-event-trigger module to define the subscription-notification smarter filter.

v01 - v02

- Introduce the group-id which allow group a set of events that can be executed together
- Change threshold trigger condition into variation trigger condition to further clarify the difference between boolean trigger condition and variation trigger condition.

- Module structure optimization.

- Usage Example Update.

v00 - v01

- Separate ietf-event-trigger.yang from Event management modeland ietf-event.yang and make it reusable in other YANG models.

- Clarify the difference between boolean trigger condition and threshold trigger condition.

- Change evt-smp-min and evt-smp-max into min-data-object and max-data-object in the data model.

Authors' Addresses

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

Email: bill.wu@huawei.com

Igor Bryskin
Individual

Email: i_bryskin@yahoo.com

Henk Birkholz
Fraunhofer SIT

Email: henk.birkholz@sit.fraunhofer.de

Xufeng Liu
Volta Networks

Email: xufeng.liu.ietf@gmail.com

Benoit Claise
Cisco

Email: bclaise@cisco.com

NETMOD Working Group
Internet-Draft
Updates: 8407 (if approved)
Intended status: Standards Track
Expires: August 23, 2021

Q. Wu
Huawei
B. Claise
Cisco
P. Liu
Z. Du
China Mobile
M. Boucadair
Orange

February 19, 2021

Self Describing Data Object Tags
draft-ietf-netmod-node-tags-01

Abstract

This document defines a method to tag data objects associated with operation and management data in YANG Modules. This YANG data object tagging method can be used to classify data objects from different YANG modules and identify characteristics data. It also can provide input, instruction, indication to selection filter and filter queries of operational state on a server during a "pub/sub" service for YANG datastore updates. When the state of all subscriptions of a particular Subscriber to be fetched is huge, the amount of data to be streamed out to the destination can be greatly reduced and only targeted to the characteristics data. These data object tags may be registered as well as assigned during the module definition; assigned by implementations; or dynamically defined and set by users.

Table of Contents

1. Introduction

As described in [I.D-ietf-netmod-module-tags], the use of tags for classification and organization is fairly ubiquitous not only within IETF protocols, but in the internet itself (e.g., "#hashtags"). A module tag defined in [I.D-ietf-netmod-module-tags] is a string associated only with a module name at module level.

At the time of writing this document (2020), there are many data models that have been specified or are being specified by various different SDOs and Open Souce community. They cover many of the networking protocols and techniques. However data objects defined by these technology specific data models might represent a portion of fault, configuration, accounting, performance, security management categories information at different locations in various different ways, lack consistent classification criteria and representation for a specific service, feature or data source.

This document defines self-describing data object tags and associates them with data objects within YANG module, which

- Provide dictionary meaning for specific targeted data objects;
- Indicate relationship between data objects within the same YANG module or from different YANG modules;
- Identify key performance metric data objects and the absolute XPath expression identifying the element path to the node;

The self describing data object tags can be used by the client to classify data objects from different YANG modules and identify characteristics data. In addition, it can provide input, instruction, indication to selection filter and filter queries of configuration or operational state on a server based on these data object tags, .e.g., return specific object type of operational state related to system-management. NETCONF clients can discover data objects with self describing data object tags supported by a NETCONF server via <get-schema> operation. The self describing data object tag capability can also be advertised via Capability Notification Model [I-D.netconf-notification-capabilities] by the NETCONF server or some place where offline document are kept. These data object tags may be registered as well as assigned during the module

definition; assigned by implementations; or dynamically defined and set by users.

This document defines a YANG module [RFC7950] which augments module tag model and provides a list of data object entries to allow for adding or removing of self describing tags as well as viewing the set of self describing tags associated with specific data objects within YANG modules.

This document defines an extension statement to be used to indicate self describing 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.

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

1.1. Self Describing Data Object Tags Use Case

1.1.1. Massive Data Object Collection

Among data object tags, the opm (object, property, metric) tags can be used to tackle massive data objects collection and only capture YANG data objects associated with performance metrics data modelled with YANG (See Figure 1).

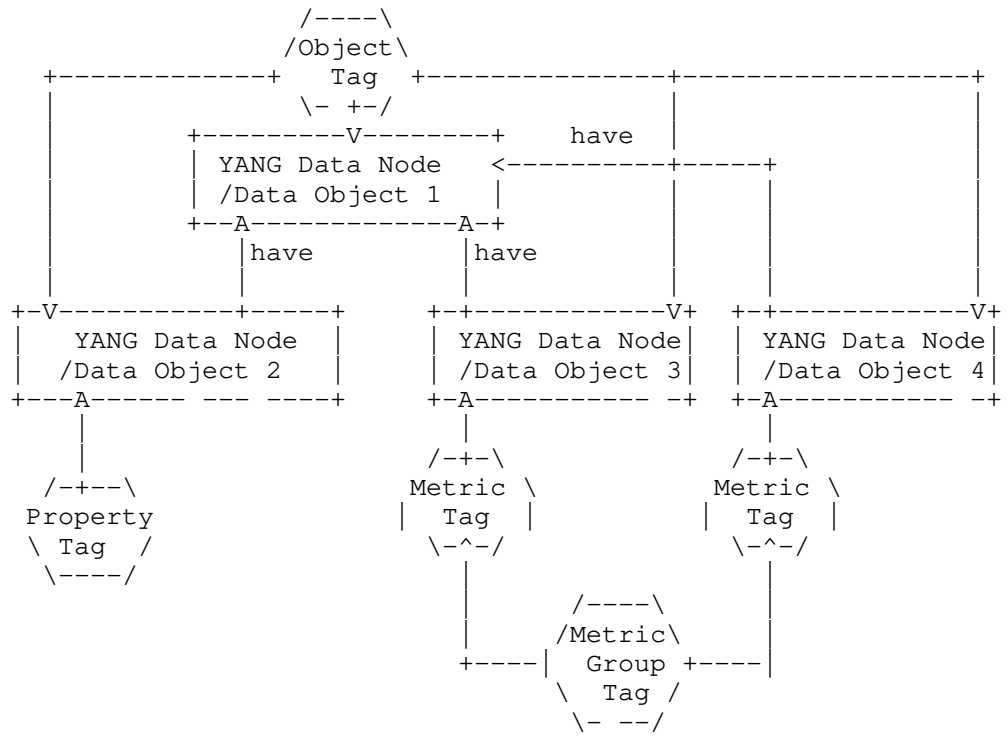


Figure 1: The Relation between Object, Property and Metric

In Figure 1, object can contain other objects called subobjects. Both object and subobjects can be modeled as YANG data nodes [RFC7950]. Object can be one of container, leaf-list and list. subobject tagged with property tag is a leaf node. subobject tagged with metric tag can be one of container, leaf-list, list, leaf node. A data Object contains one single object tag, or one single object tag and one single property tag, or one single object tag and one single metric tag. A data Object also can contain one single Metric Group tag and/or one single multi-source tag.

The use of opm tags would be to help filter discrete categories of YANG data objects scattered across the same or different YANG modules supported by a device and capture all network performance data or all property data in the single view of the truth (see Figure 2). In Figure 2, tunnel-svc data object is a container node in the tunnel-pm module and can be seen as the root object for property tagged subobjects (e.g., tunnel-svc/create-time) and metric tagged subobjects (e.g.,tunnel-svc/avg-latency). Name, create-time, modified-time are property tagged subobjects under tunnel-svc container. Avg-latency,packet -loss are metric tagged subobjects

under tunnel-svc container node. Take tunnel-svc data object and tunnel-svc/name data object as an example, tunnel-svc data object has one single object tag (i.e., ietf:object) while tunnel-svc/name data object has one object tag (ietf:object) and one property subobject tag (i.e., ietf:property). In addition, not all metric subobjects need to be tagged, e.g., only specific category (e.g., loss related) metric subobjects need to be tagged with metric-group tag which can further reduce amount data to be fetched.

Data Object	Object Tag	Property Tag	Metric Tag	Module Name
tunnel-svc	ietf: object			tunnel-pm
tunnel-svc/name	ietf: object	ietf: property		tunnel-pm
tunnel-svc/create-time	ietf: object	ietf: property		tunnel-pm
tunnel-svc/modified-time	ietf: object	ietf: property		tunnel-pm
tunnel-svc/avg-latency	ietf: object		ietf: metric	tunnel-pm
tunnel-svc/packet-loss	ietf: object		ietf: metric	tunnel-pm
tunnel-svc/min-latency	ietf: object		ietf: metric	tunnel-pm
tunnel-svc/ max-latency			ietf: metric	tunnel-pm

Figure 2: Example of OPM Tags Used in the YANG Module

If data objects in these YANG modules are suitably tagged and learnt by the client from a live server, the client can retrieve paths to all targeted data objects and then use an XPath query defined [RFC8639] [RFC8641] to list all tagged data objects which reflect network characteristics.

1.2. Terminology

1.2.1. Requirements Notation

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

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

1.2.2. Glossary

OPM - Object Property Metric

2. Data Object Tag Values

All data object tags SHOULD begin with a prefix indicating who owns their definition. An IANA registry (Section 7.1) is used to support registering data object tag prefixes. Currently 3 prefixes are defined.

No further structure is imposed by this document on the value following the registered prefix, and the value can contain any YANG type 'string' characters except carriage-returns, newlines and tabs. Therefore, designers, implementers, and users are free to add or not add any structure they may require to their own tag values.

2.1. IETF Tags Prefix

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

2.2. Vendor Tags Prefix

A vendor tag is a tag that has the prefix "vendor:". 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:vendor-defined-classifier).

2.3. User Tags Prefix

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 prefix collisions.

2.4. Reserved Tags Prefix

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. Data Object Tag Management

Tags can become associated with a data object within YANG module in a number of ways. Tags may be defined and associated at the module design time, at implementation time without the need of live server, or via user administrative control . As the main consumer of data object tags are users, users may also remove any tag from a live server, no matter how the tag became associated with a data object within a YANG module.

3.1. Module Design Tagging

A data object definition MAY indicate a set of data object tags to be added by the module implementer. These design time tags are indicated using a set of extension statements which include:

opm-tag extension statement: Classify management and operation data into object, property subobject and metric subobject three categories. Object can contain other objects called subobjects. Property and metric objects are both subobjects belonging to specific object. Both object and subobjects can be modeled as data nodes [RFC7950]. Object can be one of container, leaf-list and list. Property subobject is a leaf node. Metric subobject can be one of container, leaf-list, list, leaf. Object contains zero or many property subobjects, zero or many metric subobjects. See opm-tag example in Figure 2 and Figure 3.

metric-group extension statement: Provide meric subobjects classification (e.g., loss, jitter, delay) within the YANG module.

multi-source-tag extension statement: Identify multi-source aggregation type (e.g., aggregated, non-aggregated) related to metric subobject. 'aggregated' multi-source aggregation type allows a large number of measurements on metric subobjects from different sources of the same type (e.g.,line card, each subinterface of aggregated Ethernet interface) being combined into aggregated statistics and report as one metric subobject. 'non-aggregated' multi-source aggregation type allows measurement from each source of the same type (e.g.,line card, each subinterface of aggregated Ethernet interface) be reported separately.

Among these extension statements, the metric-group, multi-source-tag extension statements are context information related and can be used to correlate data object from the different modules.

If the data node is defined in an IETF standards track document, the data object tags MUST be IETF Tags (2.1). Thus, new data object can drive the addition of new IETF tags to the IANA registry defined in

Section 7, and the IANA registry can serve as a check against duplication.

3.2. Implementation Tagging

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

3.3. User Tagging

Data object tags of any kind, with or without a prefix, can be assigned and removed by the user from a live server using normal configuration mechanisms. In order to remove a data object tag from the operational datastore, the user adds a matching "masked-tag" entry for a given data object within the ietf-data-object-tags Module.

4. Data Object Tags Module Structure

4.1. Data Object Tags Module Tree

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

```
module: ietf-data-object-tags
  augment /tags:module-tags/tags:module:
    +--rw data-object-tags
       +--rw data-object* [object-name]
          +--rw object-name    nacm:node-instance-identifier
          +--rw tag*           tags:tag
          +--rw masked-tag*    tags:tag
```

5. YANG Module

```
<CODE BEGINS> file "ietf-data-object-tags@2019-05-03.yang"
module ietf-data-object-tags {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-data-object-tags";
  prefix ntags;

  import ietf-netconf-acm {
    prefix nacm;
  }
  import ietf-module-tags {
    prefix tags;
  }
```

```
  organization
    "IETF NetMod Working Group (NetMod)";
  contact
    "WG Web:  <https://tools.ietf.org/wg/netmod/>
     WG List: <mailto:netmod@ietf.org>
     Editor:  Qin Wu <mailto:bill.wu@huawei.com>
     Editor:  Benoit Claise <mailto:bclaise@cisco.com>
     Editor:  Peng Liu <mailto:liupengyjy@chinamobile.com>
     Editor:  Zongpeng Du <mailto:duzongpeng@chinamobile.com>
     Editor:  Mohamed Boucadair <mailto:mohamed.boucadair@orange.com>";
  description
    "This module describes a mechanism associating self-describing
     tags with YANG data object within YANG modules. Tags may be IANA
     assigned or privately defined.

     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://tools.ietf.org/html/rfcXXXX); see the RFC itself for
     full legal notices.";

  revision 2019-05-03 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: Self Describing Data Object Tags";
  }

  extension opm-tag {
    argument tag;
    description
      "The argument 'tag' is of type 'tag'. This extension statement
       is used by module authors to indicate the opm tags that SHOULD be
       added automatically by the system. Opm Tag is used to classify
       operation and management data into object, property subobject, and metric
       subobject three categories. Object can contain other objects called subobj
ects.
       Property and metric objects are both subobjects belonging to specific obje
ct.
       Both object and subobjects can be modeled as data nodes. Object can be one
 of
       container, leaf-list and list. Property subobject is a leaf node. Metric s
ubobject
       can be one of container, leaf-list, list, leaf. Object contains zero or ma
ny
```

```
       property subobjects, zero or many metric subobjects. As such the origin of
 the
       value for the pre-defined tags should be set to 'system'[RFC8342].";
  }

  extension metric-group {
    argument tag;
    description
      "The argument 'tag' is of type 'tag'.The metric-group can be
       used to provide metric subobject classification
       (e.g., loss, jitter, packet loss) within the YANG module.";
  }

  extension multi-source-tag {
    argument tag;
    description
      "The argument 'tag' is of type 'tag'.The multi-source-tag can be
       used to identify  multi-source aggregation type (e.g., aggregated,
       non-aggregated) related to metric subobject.

      'aggregated' multi-source aggregation type allows a large number of
      measurements on metric subobjects from different sources of the same
      type (e.g.,line card, each subinterface of aggregated Ethernet interface)
      being combined into aggregated statistics and report as one metric subobjec
t
      value. 'non-aggregated' multi-source aggregation type allows measurement fr
om
      each source of the same type (e.g.,line card, each subinterface of aggregat
ed
      Ethernet interface) be reported separately.";
   }

  augment "/tags:module-tags/tags:module" {
    description
      "Augment the Module Tags module with data object tag attributes";
    container data-object-tags {
      description
        "Contains the list of data objects and their associated data object tags"
;
      list data-object {
        key "object-name";
        description
          "A list of data objects and their associated data object tags";
        leaf object-name {
          type nacm:node-instance-identifier;
          mandatory true;
          description
            "The YANG data object name.";
        }
        leaf-list tag {
          type tags:tag;
          description
            "Tags associated with the data object within the YANG module. See
```

```
              the IANA 'YANG Data Object Tag Prefixes' registry for reserved
              prefixes and the IANA'IETF YANG Data Object 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 tags:tag;
            description
              "The list of tags that should not be associated with the data
               object within the YANG 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 data object within YANG module,
               but they have no operational effect.";
          }
        }
      }
    }
}
<CODE ENDS>
```

6. Guidelines to Model Writers

This section updates [RFC8407].

6.1. Define Standard Tags

A module MAY indicate, using data object tag extension statements, a set of data object tags that are to be automatically associated with data object within the module (i.e., not added through configuration).

```
module example-module-A {
       //...
       import ietf-data-node-tags { prefix ntags; }
       container top {
         ntags:opm-tag "ietf:object";
        list X {
          leaf foo {
          ntags:opm-tag "ietf:property";
           }
         }
       container Y {
         leaf bar {
         ntags:opm-tag "ietf:metric";
         }
     }
   }
 // ...
}
```

Figure 3: Data object tag example

The module writer can use existing standard data object tags, or use new data object tags defined in the data object definition, as appropriate. For IETF standardized modules, new data object tags MUST be assigned in the IANA registry defined below, see Section Section 7.2.

7. IANA Considerations

7.1. YANG Data Object Tag Prefixes Registry

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

This registry allocates tag prefixes. All YANG Data Object 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 Data Object 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 Data Object Tags Registry

IANA is asked to create 3 new registries "IETF OPM Tags","IETF Metric Group Tags","IETF Multiple Source Tags" grouped under a new "Protocol" category. These 3 registries should be included below "YANG Data Object Tag Prefixes" when listed on the same page.

3 registries allocate tags that have the registered prefix "ietf:". New values should be well considered and not achievable through a combination of already existing IETF tags.

The allocation policy for these three registries is IETF Review [RFC8126].

The initial values for these three registries are as follows.

OPM Tag	Description	Reference
ietf:object	Represent specific object type(e.g., interfaces).	[This document]
ietf:property	Represent a property subobject (e.g.,ifindex) assoiciated with specific object (e.g.,interfaces).	[This document]
ietf:metric	Represent metric subobject (e.g., ifstatistics) associated with specific object(e.g.,interfaces)	[This document]

Metric Group Tag	Description	Reference
ietf:delay	Represent the metric group which metric subobjects belong to (i.e., delay)	[This document]
ietf:jitter	Represent the metric group which metric subobjects belong to (i.e., jitter)	[This document]
ietf:loss	Represent the metric group which metric subobjects belong to (i.e.,loss)	[This document]

Multiple Source Tag	Description	Reference
ietf:non-agg	Relate to multiple source aggregation type(i.e., aggregated statistics)	[This document]
ietf:agg	Relate to multiple source aggregation type(i.e., non aggregated statistics)	[This document]

Each YANG data object can have one opm tag, zero or one metric-group tag, zero or one multi-source tag.

7.3. Updates to the IETF XML Registry

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

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

7.4. Updates to the YANG Module Names Registry

This document registers one YANG module in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registration has been made:

```
name:
   ietf-data-object-tags
namespace:
   urn:ietf:params:xml:ns:yang:ietf-data-object-tags
prefix:
   ntags
reference:
   RFC XXXX (RFC Ed.: replace XXX with actual RFC number and remove
   this note.)
```

8. Security Considerations

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

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

Users of the data object tag meta-data may define various actions to be taken based on the data object tag meta-data. These actions and their definitions are outside the scope of this document. Users will need to consider the security implications of any actions they choose to define.

9. Acknowledgements

The authors would like to thank Ran Tao for his major contributions to the initial modeling and use cases. The authors would also like to acknowledge the comments and suggestions received from Juergen Schoenwaelder, Andy Bierman, Lou Berger,Jaehoon Paul Jeong, Wei Wang, Yuan Zhang, Ander Liu, Peng Liu, YingZhen Qu, Boyuan Yan.

10. Contributors

Liang Geng
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: gengliang@chinamobile.com

11. References

11.1. Normative References

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

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

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

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

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

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

11.2. Informative References

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

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

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

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

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

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

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

Appendix A. NETCONF Example

The following is a fictional NETCONF example result from a query of the data object tags list. For the sake of brevity only a few module and associated data object 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-interfaces</t:name>
       <s:data-object-tags xmlns:s="urn:ietf:params:xml:ns:yang:ietf-data-object-
tags">
        <s:data-object>
         <s:object-name>/if:interfaces/if:interface</s:object-name>
         <s:tag>ietf:object</s:tag>
        </s:data-object>
       <s:data-object>
        <s:object-name>/if:interfaces/if:interface/if:last-change</s:object-name>
        <s:tag>ietf:property</s:tag>
       </s:data-object>
        <s:data-object>
         <s:object-name>
          /if:interfaces/if:interface/if:statistics/if:in-errors
         </s:object-name>
         <s:tag>ietf:metric</s:tag>
        </s:data-object>
      </s:data-object-tags>
    </t:module>
    <t:module>
       <t:name>ietf-ip</t:name>
       <s:data-object-tags xmlns:s="urn:ietf:params:xml:ns:yang:ietf-data-object-
tags">
       <s:data-object>
         <s:object-name>/if:interfaces/if:interface/ip:ipv4</s:object-name>
         <s:tag>ietf:object</s:tag>
       </s:data-object>
       <s:data-object>
         <s:object-name>/if:interfaces/if:interface/ip:ipv4/ip:enable</s:object-n
ame>
         <s:tag>ietf:property</s:tag>
       </s:data-object>
        <s:data-object>
        <s:object-name>/if:interfaces/if:interface/ip:ipv4/ip:mtu</s:object-name>
        <s:tag>ietf:metric</s:tag>
       </s:data-object>
      </s:data-object-tags>
    </t:module>
  </t:module-tags>
</ns0:data>
```

Appendix B. 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-data-object-tags-state@2019-05-03.yang"
module ietf-data-object-tags-state {
```

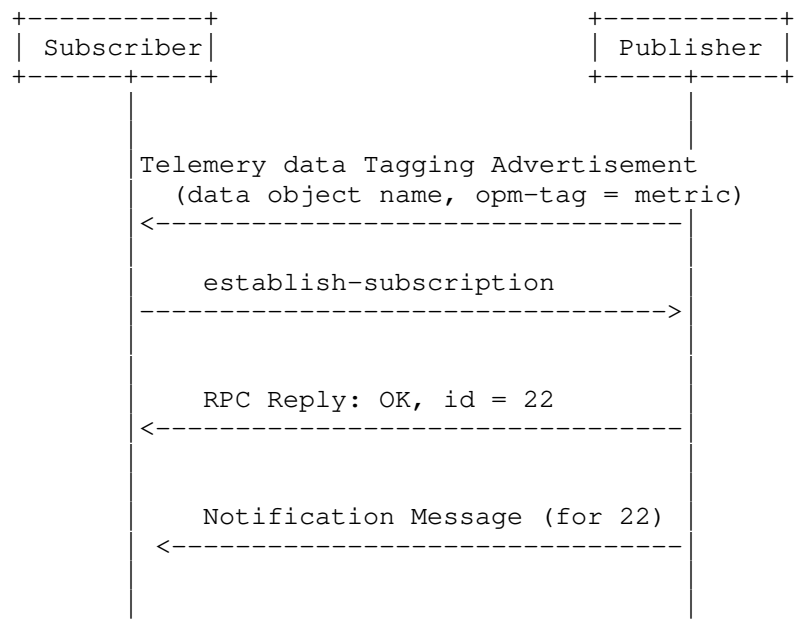
```
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-data-object-tags-state";
  prefix ntags-s;

  import ietf-netconf-acm {
    prefix nacm;
  }
  import ietf-module-tags {
    prefix tags;
  }
  organization
    "IETF NetMod Working Group (NetMod)";
  contact
    "WG Web:  <https://tools.ietf.org/wg/netmod/>
     WG List: <mailto:netmod@ietf.org>
     Editor:  Qin Wu <mailto:bill.wu@huawei.com>
     Editor:  Benoit Claise <mailto:bclaise@cisco.com>
     Editor:  Peng Liu <mailto:liupengyjy@chinamobile.com>
     Editor:  Zongpeng Du <mailto:duzongpeng@chinamobile.com>";
  description
    "This module describes a mechanism associating self-describing
     tags with YANG data object within YANG modules. Tags may be IANA
     assigned or privately defined.

     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://tools.ietf.org/html/rfcXXXX); see the RFC itself for
     full legal notices.";

  revision 2019-05-03 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: Self Describing Data Object Tags";
  }

  extension opm-tag {
    argument tag;
    description
```

```
      "The argument 'tag' is of type 'tag'. This extension statement
       is used by module authors to indicate the opm tags that SHOULD be
       added automatically by the system. Opm Tag is used to classify
       operation and management data into object, property subobject, and metric
       subobject three categories. Object can contain other objects called subobj
ects.
       Property and metric objects are both subobjects belonging to specific obje
ct.
       Both object and subobjects can be modeled as data nodes. Object can be one
 of
       container, leaf-list and list. Property subobject is a leaf node. Metric s
ubobject
       can be one of container, leaf-list, list, leaf. Object contains zero or ma
ny
       property subobjects, zero or many metric subobjects. As such the origin of
 the value
       for the pre-defined tags should be set to 'system'[RFC8342].";
  }
  extension metric-group {
    argument tag;
    description
      "The argument 'tag' is of type 'tag'.The metric-group can be
       used to provide metric subobject classification
       (e.g., loss, jitter, packet loss)within the YANG module.";
  }
  extension multi-source-tag {
    argument tag;
    description
      "The argument 'tag' is of type 'tag'.The multi-source-tag can be
       used to identify  multi-source aggregation type (e.g., aggregated,
       non-aggregated) related to metric subobject.

      'aggregated' multi-source aggregation type allows a large number of
      measurements on metric subobjects from different sources of the same
      type (e.g.,line card, each subinterface of aggregated Ethernet interface)
      being combined into aggregated statistics and report as one metric subobjec
t
      value. 'non-aggregated' multi-source aggregation type allows measurement fr
om
      each source of the same type (e.g.,line card, each subinterface of aggregat
ed
      Ethernet interface) be reported separately.";
   }

  augment "/tags:module-tags/tags:module" {
    description
      "Augment the Module Tags module with data object tag attributes.";
    container data-object-tags {
      config false;
      status deprecated;
      description
        "Contains the list of data objects and their associated self describing t
ags.";
      list data-object {
        key "object-name";
        status deprecated;
        description
          "A list of data objects and their associated self describing tags.";
```

```
        leaf object-name {
          type nacm:node-instance-identifier;
          mandatory true;
          status deprecated;
          description
            "The YANG data object name.";
        }
        leaf-list tag {
          type tags:tag;
          status deprecated;
          description
            "Tags associated with the data object within the YANG module. See
             the IANA 'YANG Data Object Tag Prefixes' registry for reserved
             prefixes and the IANA'IETF YANG Data Object 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 tags:tag;
          status deprecated;
          description
            "The list of tags that should not be associated with the data
             object within the YANG 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 data object within YANG module,
             but they have no operational effect.";
        }
      }
    }
  }
}
<CODE ENDS>
```

Appendix C. Targeted data object collection example

The following subsections provides targeted data object collection example which helps reduce amount of data to be fetched. The subscription "id" values of 22 used below is just an example. In production, the actual values of "id" might not be small integers.



The publisher advertises telemetry data object capability to the subscriber to instruct the receiver to subscribe tagged data object (e.g., performance metric data object) using standard subscribed notification mechanism [RFC8639].

The following XML example [W3C.REC-xml-20081126] illustrates the advertisment of the list of available target objects using YANG instance file format [I-D.ietf-netmod-yang-instance-file-format]:

```
<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=\
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-router-notification-capabilities</name>
  <content-schema>
    <module>ietf-system-capabilities@2020-03-23</module>
    <module>ietf-notification-capabilities@2020-03-23</module>
    <module>ietf-data-export-capabilities@2020-03-23</module>
  </content-schema>
  <!-- revision date, contact, etc. -->
  <description>Defines the notification capabilities of an acme-router.
    The router only has running, and operational datastores.
    Every change can be reported on-change from running, but
    only config=true nodes and some config=false data from operational.
    Statistics are not reported based on timer based trigger and counter
    threshold based trigger.
  </description>
  <content-data>
    <system-capabilities \
      xmlns="urn:ietf:params:xml:ns:yang:ietf-system-capabilities" \
      xmlns:inc=\
        "urn:ietf:params:xml:ns:yang:ietf-notification-capabilities" \
      xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      <datastore-capabilities>
        <datastore>ds:operational</datastore>
        <per-node-capabilities>
          <node-selector>\
              /if:interfaces/if:interface/if:statistics/if:in-errors\
          </node-selector>
          <sec:self-describing-capabilities>
            <sec:opm-tag>metric</sec:opm-tag>
            <sec:metric-group>loss</sec:metric-group>
          </sec:self-describing-capabilities>
        </per-node-capabilities>
      </datastore-capabilities>
    </system-capabilities>
  </content-data>
</instance-data-set>
```

With telemetry data tagging information carried in the Telemetry data Tagging Advertisement, the subscriber identifies targeted data object and associated data path to the datastore node and sends a standard establish-subscription RPC [RFC8639] to subscribe tagged data objects that are interests to the client application from the publisher.

```
 <netconf:rpc message-id="101"
     xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
   <establish-subscription
       xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
       xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
     <yp:datastore
         xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
       ds:operational
     </yp:datastore>
     <yp:datastore-xpath-filter
        xmlns:ex="https://example.com/sample-data/1.0">
       /if:interfaces/if:interface/if:statistics/if:in-errors
     </yp:datastore-xpath-filter>
     <yp:periodic>
       <yp:period>500</yp:period>
     </yp:periodic>
   </establish-subscription>
 </netconf:rpc>
```

The publisher returns specific object type of operational state (e.g., in-errors statistics data) subscribed by the client.

Appendix D. Changes between Revisions

v00 - v01

- Merge self describing data object tag use case section into introduction section as a subsection;
- Add one glossary section;
- Clarify the relation between data object, object tag, property tag and metric tag in Self Describing Data Object Tags Use Case section;
- Add update to RFC8407 in the front page.

Authors' Addresses

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

Email: bill.wu@huawei.com

Benoit Claise
Cisco
De Kleetlaan 6a b1
Diegem 1831
Belgium

Email: bclaise@cisco.com

Peng Liu
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: liupengyjy@chinamobile.com

Zongpeng Du
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: duzongpeng@chinamobile.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

NETMOD Working Group
Internet-Draft
Updates: 8407 (if approved)
Intended status: Standards Track
Expires: November 14, 2021

Q. Wu
B. Claise
Huawei
P. Liu
Z. Du
China Mobile
M. Boucadair
Orange

May 13, 2021

Self Describing Data Object Tags
draft-ietf-netmod-node-tags-03

Abstract

This document defines a method to tag data objects associated with operation and management data in YANG modules. This YANG data object tagging method can be used to classify data objects from different YANG modules and identify their characteristics data. It can also provide input, instruction, indication to selection filter, and filter queries of operational state on a server during a "pub/sub" service for YANG datastore updates. When the subscriptions of a particular subscriber to be fetched is very large, the amount of data to be streamed out to the destination can be reduced and only targeted to the characteristics data. These data object tags may be registered as well as assigned during the module definition, assigned by implementations, or dynamically defined and set by users.

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 14, 2021.

Table of Contents

1. Introduction

As described in [RFC8819], the use of tags for classification and organization is fairly ubiquitous not only within IETF protocols, but in the Internet itself (e.g., "#hashtags"). As a reminder, a module tag defined in [I.D-ietf-netmod-module-tags] is a string associated only with a module name at the module level.

At the time of writing this document (2020), there are many data models that have been specified or are being specified by various SDOs and the Open Source community. These models cover many of the networking protocols and techniques. However, data objects defined by these technology-specific data models might represent a portion of fault, configuration, accounting, performance, and security management categories information at different locations in various different ways. Let alone the lack consistent classification criteria and representation for a specific service, feature, or data source.

This document defines self-describing data object tags and associates them with data objects within a YANG module, which:

- Provide dictionary meaning for specific targeted data objects.
- Indicate relationship between data objects within the same YANG module or from different YANG modules.
- Identify key performance metric data objects and the absolute XPath expression identifying the element path to the node.

The self describing data object tags can be used by the NETCONF/ RESTCONF client to classify data objects from different YANG modules and identify characteristics data. In addition, it can provide input, instruction, indication to selection filter and filter queries of configuration or operational state on a server based on these data object tags, e.g., return specific object type of operational state related to system-management. NETCONF clients can discover data objects with self describing data object tags supported by a NETCONF server by means of <get-schema> operation. The self describing data object tag capability can also be advertised using the capability notification model [I-D.netconf-notification-capabilities] by the NETCONF server or some place where offline document are kept. These

data object tags may be registered or assigned during the module definition, assigned by implementations, or dynamically defined and set by users.

This document defines a YANG module [RFC7950] which augments the module tag model and provides a list of data object entries to allow for adding or removing of self describing tags as well as viewing the set of self describing tags associated with specific data objects within YANG modules.

This document defines an extension statement to be used to indicate self describing 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 9).

Section 8 provides guidelines for authors of YANG data models.

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

The meaning of the symbols can be found in [RFC8340].

2. Terminology

2.1. Requirements Notation

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

2.2. Glossary

OPM Object Property Metric

3. Self Describing Data Object Tags: Massive Data Object Collection Use Case

Among data object tags, the 'opm' (object, property, metric) tags can be used to tackle massive data objects collection and only capture YANG data objects associated with performance metrics data modelled with YANG (Figure 1).

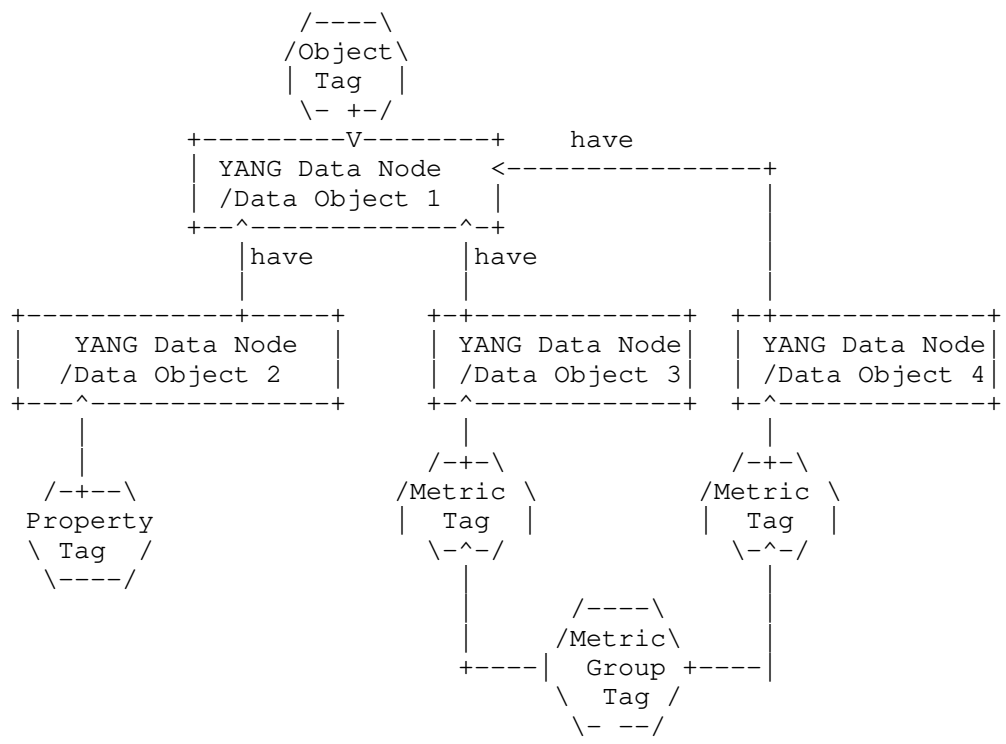


Figure 1: The Relation between Object, Property and Metric

In Figure 1, data objects can contain other data objects called subobjects. Both object and subobjects can be modeled as YANG data nodes [RFC7950]. Data objects that contain other data objects can be one of 'container', 'leaf-list', and 'list' and are tagged with object tag. A subobject tagged with the property tag is a 'leaf' node. Subobjects tagged with the metric tag can be one of 'container', 'leaf-list', 'list', or 'leaf' data node.

A data object contains one single object tag, one single property tag, or one single metric tag. A data object tagged with metric tag also can have one or multiple Metric Type tags and/or one single multi-source tag.

The use of 'opm' tags is meant to help filter discrete categories of YANG data objects scattered across the same or different YANG modules supported by a device and capture all network performance data or all property data in the single view of the data. In Figure 2, 'tunnel-svc' data object is a container node defined in the 'tunnel-pm' module and can be seen as the root object for property tagged subobjects (e.g., 'tunnel-svc'/'create-time') and metric tagged

subobjects (e.g., 'tunnel-svc'/'avg-latency'). The 'name', 'create-time', and 'modified-time' are property tagged subobjects under 'tunnel-svc' container. The 'avg-latency' and 'packet-loss' metrics are tagged subobjects under 'tunnel-svc' container node. Consider 'tunnel-svc' data object and tunnel-svc/name data object as an example, 'tunnel-svc' data object has one single object tag (i.e., 'ietf:object') while tunnel-svc/name data object has one property subobject tag (i.e., 'ietf:property'). In addition, not all metric subobjects need to be tagged, e.g., only specific category such as loss-related metric subobjects need to be tagged with metric-type tag which can further reduce amount data to be fetched.

Data Object	Object Tag	Property Tag	Metric Tag	Module Name
tunnel-svc	ietf: object			tunnel-pm
tunnel-svc/name		ietf: property		tunnel-pm
tunnel-svc/create-time		ietf: property		tunnel-pm
tunnel-svc/modified-time		ietf: property		tunnel-pm
tunnel-svc/avg-latency			ietf: metric	tunnel-pm
tunnel-svc/packet-loss			ietf: metric	tunnel-pm
tunnel-svc/min-latency			ietf: metric	tunnel-pm
tunnel-svc/ max-latency			ietf: metric	tunnel-pm

Figure 2: Example of OPM Tags Used in the YANG Module

If data objects in YANG modules are suitably tagged and learnt by the client from a live server, the client can retrieve paths to all targeted data objects and then use an XPath query defined [RFC8639][RFC8641] to list all tagged data objects which reflect network characteristics.

4. Data Object Tag Values

All data object tags SHOULD begin with a prefix indicating who owns their definition. To that aim, an IANA registry (Section 9.1) is

used to support registering data object tag prefixes. Three prefixes are defined in the following subsections.

No further structure is imposed by this specification on the value following the registered prefix other than the value can contain any YANG type 'string' characters except carriage-returns, newlines, and tabs. Therefore, designers, implementers, and users are free to add or not any structure they may require to their own tag values.

4.1. IETF Tags Prefix

An IETF tag is a data object tag that has the prefix "ietf:".

All IETF data object tags are registered with IANA in a registry defined Section 9.2.

4.2. Vendor Tags Prefix

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 includes extra identification in the tag to avoid collisions such as using the enterprise or organization name following the "vendor:" prefix (e.g., vendor:vendor-defined-classifier).

4.3. User Tags Prefix

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 prefix collisions.

4.4. Reserved Tags Prefix

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).

5. Data Object Tag Management

Tags may be associated with a data object within a YANG module in a number of ways. Typically, tags may be defined and associated at the module design time, at implementation time without the need of live server, or via user administrative control. As the main consumer of data object tags are users, users may also remove any tag from a live

server, no matter how the tag became associated with a data object within a YANG module.

5.1. Module Design Tagging

A data object definition MAY indicate a set of data object tags to be added by a module's implementer. These design time tags are indicated using a set of extension statements which include:

opm-tag extension statement: Classifies management and operation data into object, property subobject, and metric subobject categories. Both object and subobjects can be modeled as YANG data nodes [RFC7950]. Data objects that contain other data objects can be one of 'container', 'leaf-list', and 'list' and are tagged with object tag. A subobject tagged with the property tag is a 'leaf' node. Subobjects tagged with the metric tag can be one of 'container', 'leaf-list', 'list', or 'leaf' data node. A data object contains one single object tag, one single property tag, or one single metric tag. A data object tagged with metric tag also can have one or multiple Metric type tag and/or one single multi-source tag. See the examples depicted in Figure 2 and Figure 3.

metric-type extension statement: Provides metric data objects classification (e.g., loss, jitter, delay, counter, gauge, histogram, summary, unknown) within the YANG module.

multi-source-tag extension statement: Identifies multi-source aggregation type (e.g., aggregated, non-aggregated) related to metric subobject. 'aggregated' multi-source aggregation type allows a large number of measurements on metric subobjects from different sources of the same type (e.g., line card, each subinterface of aggregated Ethernet interface) being combined into aggregated statistics and report as one metric subobject. 'non-aggregated' multi-source aggregation type allows measurement from each source of the same type (e.g., line card, each subinterface of aggregated Ethernet interface) be reported separately.

Among these extension statements, the metric-type and multi-source-tag extension statements are context information that can be used to correlate data object from the different modules.

If the data node is defined in an IETF standards track document, the data object tags MUST be IETF Tags (Section 4.1). Thus, new data object can drive the addition of new IETF tags to the IANA registry defined in Section 9, and the IANA registry can serve as a check against duplication.

5.2. Implementation Tagging

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

5.3. User Tagging

Data object tags of any kind, with or without a prefix, can be assigned and removed by the user from a live server using normal configuration mechanisms. In order to remove a data object tag from the operational datastore, the user adds a matching "masked-tag" entry for a given data object within the 'ietf-data-object-tags' module.

6. Data Object Tags Module Structure

6.1. Data Object Tags Module Tree

The tree associated with the "ietf-data-object-tags" module is as follows:

```
module: ietf-data-object-tags
  augment /tags:module-tags/tags:module:
    +--rw data-object-tags
       +--rw data-object* [object-name]
          +--rw object-name    nacm:node-instance-identifier
          +--rw tag*           tags:tag
          +--rw masked-tag*    tags:tag
```

7. YANG Module

```
<CODE BEGINS> file "ietf-data-object-tags@2021-05-03.yang"
module ietf-data-object-tags {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-data-object-tags";
  prefix ntags;

  import ietf-netconf-acm {
    prefix nacm;
  }
  import ietf-module-tags {
    prefix tags;
  }

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

```
    "WG Web:  <https://tools.ietf.org/wg/netmod/>
     WG List: <mailto:netmod@ietf.org>
     Editor:  Qin Wu <mailto:bill.wu@huawei.com>
     Editor:  Benoit Claise <mailto:bclaise@cisco.com>
     Editor:  Peng Liu <mailto:liupengyjy@chinamobile.com>
     Editor:  Zongpeng Du <mailto:duzongpeng@chinamobile.com>
     Editor:  Mohamed Boucadair <mailto:mohamed.boucadair@orange.com>";
  description
    "This module describes a mechanism associating self-describing
     tags with YANG data object within YANG modules. Tags may be IANA
     assigned or privately defined.

     Copyright (c) 2021 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://tools.ietf.org/html/rfcXXXX); see the RFC itself for
     full legal notices.";

  revision 2021-05-03 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: Self Describing Data Object Tags";
  }

  extension opm-tag {
    argument tag;
    description
      "The argument 'tag' is of type 'tag'. This extension statement
       is used by module authors to indicate the opm tags that should
       be added automatically by the system. Opm Tag is used to
       classify operation and management data object into object,
       property, and metric three categories. Data Object can contain
       other data objects called subobjects. Both object and subobjects
       can be modeled as data nodes. The Data Object tagged with object
       tag can be one of container, leaf-list and list. Data Object
       tagged with the Property tag is a leaf node. Data Object tagged with
       the Metric tag can be one of container, leaf-list, list, leaf.
       Data objects tagged with either property tag or metric tag are
       subobjects belonging to specific root data object.  Data Object
```

```
       contains One object tag or one property tag, or one metric tag. As
       such the origin of the value for the pre-defined tags should be set
       to 'system'.";
  }

  extension metric-type {
    argument tag;
    description
      "The argument 'tag' is of type 'tag'. The metric type can be
       used to provide metric data object classification
       (e.g., loss, jitter, packet loss,counter, gauge, histogram,
       summary, unknown) within the YANG module.";
  }

  extension multi-source-tag {
    argument tag;
    description
      "The argument 'tag' is of type 'tag'. The multi-source-tag can be
       used to identify  multi-source aggregation type (e.g., aggregated,
       non-aggregated) related to metric subobject.

      'aggregated' multi-source aggregation type allows a large number of
      measurements on metric subobjects from different sources of the same
      type (e.g., line card, each subinterface of aggregated Ethernet interface)
      being combined into aggregated statistics and report as one metric subobjec
t
      value. 'non-aggregated' multi-source aggregation type allows measurement fr
om
      each source of the same type (e.g., line card, each subinterface of aggrega
ted
      Ethernet interface) be reported separately.";
   }

  augment "/tags:module-tags/tags:module" {
    description
      "Augment the Module Tags module with data object tag attributes";
    container data-object-tags {
      description
        "Contains the list of data objects and their associated data object tags"
;
      list data-object {
        key "object-name";
        description
          "A list of data objects and their associated data object tags";
        leaf object-name {
          type nacm:node-instance-identifier;
          mandatory true;
          description
            "The YANG data object name.";
        }
        leaf-list tag {
          type tags:tag;
```

```
          description
            "Tags associated with the data object within the YANG module. See
             the IANA 'YANG Data Object Tag Prefixes' registry for reserved
             prefixes and the IANA 'IETF YANG Data Object Tags' registry for
             IETF tags.

             The 'operational' state 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 tags:tag;
          description
            "The list of tags that should not be associated with the data
             object within the YANG module. The user can remove (mask) tags from
the
             operational state datastore by adding them to
             this list. It is not an error to add tags to this list
             that are not associated with the data object within YANG module,
             but they have no operational effect.";
        }
      }
    }
  }
}
<CODE ENDS>
```

8. Guidelines to Model Writers

This section updates [RFC8407].

8.1. Define Standard Tags

A module MAY indicate, using data object tag extension statements, a set of data object tags that are to be automatically associated with data object within the module (i.e., not added through configuration).

```
module example-module-A {
       //...
       import ietf-data-node-tags { prefix ntags; }
       container top {
         ntags:opm-tag "ietf:object";
        list X {
          leaf foo {
          ntags:opm-tag "ietf:property";
           }
         }
       container Y {
         leaf bar {
         ntags:opm-tag "ietf:metric";
         }
     }
   }
 // ...
}
```

Figure 3: Data object tag example

The module writer can use existing standard data object tags, or use new data object tags defined in the data object definition, as appropriate. For IETF standardized modules, new data object tags MUST be assigned in the IANA registry defined below, see Section Section 9.2.

9. IANA Considerations

9.1. YANG Data Object Tag Prefixes Registry

This document requests IANA to create a new registry entitled "YANG Data Object Tag Prefixes" grouped under a new "Protocol" category named "YANG Data Object Tag Prefixes".

This registry allocates tag prefixes. All YANG Data Object 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 Data Object Tags registry	[This document]	IETF
vendor:	Non-registered tags allocated by the module's 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.

9.2. IETF YANG Data Object Tags Registry

This document requests IANA to create three new registries "IETF OPM Tags","IETF Metric Type Tags","IETF Multiple Source Tags" grouped under a new "Protocol" category. These 3 registries should be included below "YANG Data Object Tag Prefixes" when listed on the same page.

Three registries are used to allocate tags that have the registered prefix "ietf:". New values should be well considered and not achievable through a combination of already existing IETF tags.

The allocation policy for these three registries is IETF Review [RFC8126].

The initial values for these three registries are as follows:

OPM Tag	Description	Reference
ietf:object	Represents Root object containing other data objects (e.g., interfaces)	[This document]
ietf:property	Represents a property data object(e.g., ifindex) associated with a specific root object (e.g., interfaces)	[This document]

ietf:metric	Represent metric data object(e.g., ifstatistics) associated with specific root object(e.g., interfaces)	[This document]

Metric Type Tag	Description	Reference
ietf:delay	Represents the delay metric group to which the metric data objects belong to.	[This document]
ietf:jitter	Represents the jitter metric group to which the metric data objects belong to.	[This document]
ietf:loss	Represents the loss metric group to which the metric data objects belong to.	[This document]
ietf:counter	Represents any metric value associated with a metric data object that monotonically increases over time, starting from zero.	[This document]
ietf:gauge	Represents current measurements associated with a metric data object that may increase, decrease or stay constant.	[This document]
ietf:histogram	Represents the frequency of value observations associated with a metric data object that falls into specific predefined range.	[This document
ietf:histogram	Represents the metric value associated with a metric data object that measures distributions of discrete events without knowing predefined range.	[This document
ietf:unknown	Represents the metric value associated with metric	[This document

	data object that can not determine the type of metric.	

Multiple Source Tag	Description	Reference
ietf:agg	Relates to multiple sources aggregation type (i.e., aggregated statistics)	[This document]
ietf:non-agg	Relates to multiple sources aggregation type (i.e., non-aggregated statistics)	[This document]

Each YANG data object can have one 'opm' tag, zero or one metric-type tag, zero or one multi-source tag.

9.3. Updates to the IETF XML Registry

This document requests IANA to register a new URI in the "IETF XML Registry" [RFC3688]. Following the format in [RFC3688], the following registration has been made:

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

9.4. Updates to the YANG Module Names Registry

This document requests IANA to register one YANG module in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registration has been made:

```
name: ietf-data-object-tags
namespace: urn:ietf:params:xml:ns:yang:ietf-data-object-tags
prefix: ntags
reference: RFC XXXX
maintained by IANA: N
```

10. Security Considerations

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

is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

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

Users of the data object tag meta-data may define various actions to be taken based on the data object 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.

11. Acknowledgements

The authors would like to thank Ran Tao for his major contributions to the initial modeling and use cases. The authors would also like to acknowledge the comments and suggestions received from Juergen Schoenwaelder, Andy Bierman, Lou Berger,Jaehoon Paul Jeong, Wei Wang, Yuan Zhang, Ander Liu, Peng Liu, YingZhen Qu, Boyuan Yan.

12. Contributors

Liang Geng
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: gengliang@chinamobile.com

13. References

13.1. Normative References

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

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

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

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

[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>.

[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>.

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

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

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

[RFC8819] Hopps, C., Berger, L., and D. Bogdanovic, "YANG Module Tags", RFC 8819, DOI 10.17487/RFC8819, January 2021, <https://www.rfc-editor.org/info/rfc8819>.

13.2. Informative References

[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>.

[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>.

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

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

Appendix A. NETCONF Example

The following is a NETCONF example result from a query of the data object tags list. For the sake of brevity only a few module and associated data object results are provided.

```
<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-interfaces</t:name>
       <s:data-object-tags xmlns:s="urn:ietf:params:xml:ns:yang:ietf-data-object-
tags">
        <s:data-object>
         <s:object-name>/if:interfaces/if:interface</s:object-name>
         <s:tag>ietf:object</s:tag>
        </s:data-object>
       <s:data-object>
        <s:object-name>/if:interfaces/if:interface/if:last-change</s:object-name>
        <s:tag>ietf:property</s:tag>
       </s:data-object>
        <s:data-object>
         <s:object-name>
          /if:interfaces/if:interface/if:statistics/if:in-errors
         </s:object-name>
         <s:tag>ietf:metric</s:tag>
        </s:data-object>
      </s:data-object-tags>
    </t:module>
    <t:module>
       <t:name>ietf-ip</t:name>
       <s:data-object-tags xmlns:s="urn:ietf:params:xml:ns:yang:ietf-data-object-
tags">
       <s:data-object>
         <s:object-name>/if:interfaces/if:interface/ip:ipv4</s:object-name>
         <s:tag>ietf:object</s:tag>
       </s:data-object>
       <s:data-object>
         <s:object-name>/if:interfaces/if:interface/ip:ipv4/ip:enable</s:object-n
ame>
         <s:tag>ietf:property</s:tag>
       </s:data-object>
        <s:data-object>
        <s:object-name>/if:interfaces/if:interface/ip:ipv4/ip:mtu</s:object-name>
        <s:tag>ietf:metric</s:tag>
       </s:data-object>
      </s:data-object-tags>
    </t:module>
  </t:module-tags>
</ns0:data>
```

Appendix B. 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-data-object-tags-state@2021-05-03.yang"
module ietf-data-object-tags-state {
```

```
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-data-object-tags-state";
  prefix ntags-s;

  import ietf-netconf-acm {
    prefix nacm;
  }
  import ietf-module-tags {
    prefix tags;
  }
  organization
    "IETF NetMod Working Group (NetMod)";
  contact
    "WG Web:  <https://tools.ietf.org/wg/netmod/>
     WG List: <mailto:netmod@ietf.org>
     Editor:  Qin Wu <mailto:bill.wu@huawei.com>
     Editor:  Benoit Claise <mailto:bclaise@cisco.com>
     Editor:  Peng Liu <mailto:liupengyjy@chinamobile.com>
     Editor:  Zongpeng Du <mailto:duzongpeng@chinamobile.com>
     Editor:  Mohamed Boucadair <mailto:mohamed.boucadair@orange.com>";
  description
    "This module describes a mechanism associating self-describing
     tags with YANG data object within YANG modules. Tags may be IANA
     assigned or privately defined.

     Copyright (c) 2021 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://tools.ietf.org/html/rfcXXXX); see the RFC itself for
     full legal notices.";

  revision 2021-05-03 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: Self Describing Data Object Tags";
  }

  extension opm-tag {
    argument tag;
```

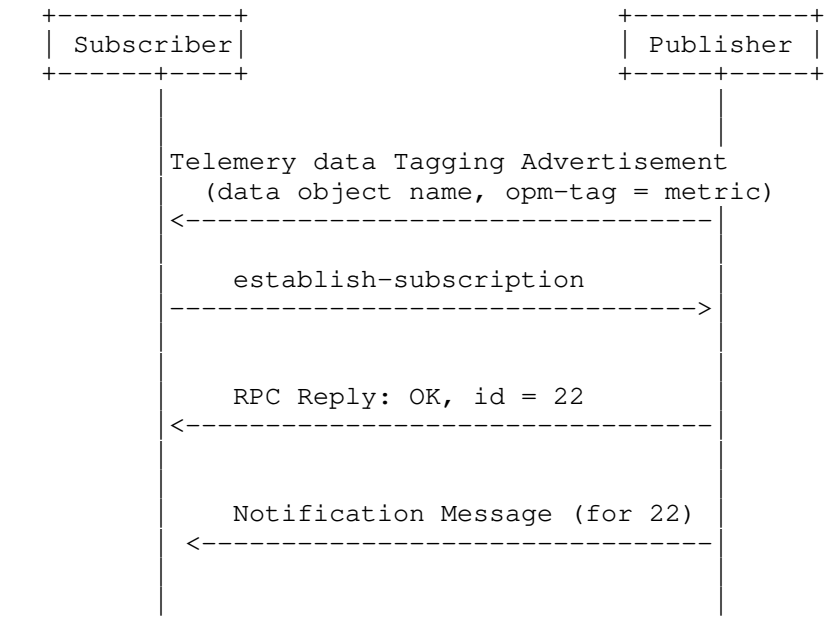
```
    description
      "The argument 'tag' is of type 'tag'. This extension statement
       is used by module authors to indicate the opm tags that should be
       added automatically by the system. Opm Tag is used to classify
       operation and management data into object, property subobject, and metric
       subobject three categories. Object can contain other objects called subobj
ects.
       Property and metric objects are both subobjects belonging to specific obje
ct.
       Both object and subobjects can be modeled as data nodes. Object can be one
 of
       container, leaf-list and list. Property subobject is a leaf node. Metric s
ubobject
       can be one of container, leaf-list, list, leaf. Object contains zero or ma
ny
       property subobjects, zero or many metric subobjects. As such the origin of
 the value
       for the pre-defined tags should be set to 'system'.";
  }
  extension metric-type {
    argument tag;
    description
      "The argument 'tag' is of type 'tag'.The metric-type can be
       used to provide metric subobject classification
       (e.g., loss, jitter, packet loss, guage, counter, histogram, unknow, etc.)
       within the YANG module.";
  }
  extension multi-source-tag {
    argument tag;
    description
      "The argument 'tag' is of type 'tag'.The multi-source-tag can be
       used to identify  multi-source aggregation type (e.g., aggregated,
       non-aggregated) related to metric subobject.

      'aggregated' multi-source aggregation type allows a large number of
      measurements on metric subobjects from different sources of the same
      type (e.g., line card, each subinterface of aggregated Ethernet interface)
      being combined into aggregated statistics and report as one metric subobjec
t
      value. 'non-aggregated' multi-source aggregation type allows measurement fr
om
      each source of the same type (e.g., line card, each subinterface of aggrega
ted
      Ethernet interface) be reported separately.";
   }

  augment "/tags:module-tags/tags:module" {
    description
      "Augment the Module Tags module with data object tag attributes.";
    container data-object-tags {
      config false;
      status deprecated;
      description
        "Contains the list of data objects and their associated self describing t
ags.";
      list data-object {
        key "object-name";
        status deprecated;
```

```
        description
          "A list of data objects and their associated self describing tags.";
        leaf object-name {
          type nacm:node-instance-identifier;
          mandatory true;
          status deprecated;
          description
            "The YANG data object name.";
        }
        leaf-list tag {
          type tags:tag;
          status deprecated;
          description
            "Tags associated with the data object within the YANG module. See
             the IANA 'YANG Data Object Tag Prefixes' registry for reserved
             prefixes and the IANA 'IETF YANG Data Object Tags' registry for
             IETF tags.

             The 'operational' state 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 tags:tag;
          status deprecated;
          description
            "The list of tags that should not be associated with the data
             object within the YANG module. The user can remove (mask) tags from
the
             operational state datastore by adding them to
             this list. It is not an error to add tags to this list
             that are not associated with the data object within YANG module,
             but they have no operational effect.";
        }
      }
    }
  }
}
<CODE ENDS>
```

Appendix C. Targeted data object collection example

The following provides targeted data object collection example which helps reduce amount of data to be fetched. The subscription "id"

values of 22 used below is just an example. In production, the actual values of "id" might not be small integers.



The publisher advertises telemetry data object capability to the subscriber to instruct the receiver to subscribe tagged data object (e.g., performance metric data object) using standard subscribed notification mechanism [RFC8639].

The following XML example [W3C.REC-xml-20081126] illustrates the advertisment of the list of available target objects using YANG instance file format [I-D.ietf-netmod-yang-instance-file-format]:

```
<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=\
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-router-notification-capabilities</name>
  <content-schema>
    <module>ietf-system-capabilities@2020-03-23</module>
    <module>ietf-notification-capabilities@2020-03-23</module>
    <module>ietf-data-export-capabilities@2020-03-23</module>
  </content-schema>
  <!-- revision date, contact, etc. -->
  <description>Defines the notification capabilities of an acme-router.
    The router only has running, and operational datastores.
    Every change can be reported on-change from running, but
    only config=true nodes and some config=false data from operational.
    Statistics are not reported based on timer based trigger and counter
    threshold based trigger.
  </description>
  <content-data>
    <system-capabilities \
      xmlns="urn:ietf:params:xml:ns:yang:ietf-system-capabilities" \
      xmlns:inc=\
        "urn:ietf:params:xml:ns:yang:ietf-notification-capabilities" \
      xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      <datastore-capabilities>
        <datastore>ds:operational</datastore>
        <per-node-capabilities>
          <node-selector>\
              /if:interfaces/if:interface/if:statistics/if:in-errors\
          </node-selector>
          <sec:self-describing-capabilities>
            <sec:opm-tag>metric</sec:opm-tag>
            <sec:metric-type>loss</sec:metric-type>
          </sec:self-describing-capabilities>
        </per-node-capabilities>
      </datastore-capabilities>
    </system-capabilities>
  </content-data>
</instance-data-set>
```

With telemetry data tagging information carried in the Telemetry data Tagging Advertisement, the subscriber identifies targeted data object and associated data path to the datastore node and sends a standard establish-subscription RPC [RFC8639] to subscribe tagged data objects that are interests to the client application from the publisher.

```
 <netconf:rpc message-id="101"
     xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
   <establish-subscription
       xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
       xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
     <yp:datastore
         xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
       ds:operational
     </yp:datastore>
     <yp:datastore-xpath-filter
        xmlns:ex="https://example.com/sample-data/1.0">
       /if:interfaces/if:interface/if:statistics/if:in-errors
     </yp:datastore-xpath-filter>
     <yp:periodic>
       <yp:period>500</yp:period>
     </yp:periodic>
   </establish-subscription>
 </netconf:rpc>
```

The publisher returns specific object type of operational state (e.g., in-errors statistics data) subscribed by the client.

Appendix D. Changes between Revisions

v02 - v03

- Additional Editorial changes.
- Security section enhancement.
- Nits fixed.

v01 - v02

- Clarify the relation between data object, object tag, property tag and metric tag in figure 1 and figure 2 and related description;
- Change Metric Group into Metric Type in the YANG model;
- Add 5 metric types in section 7.2;

v00 - v01

- Merge self describing data object tag use case section into introduction section as a subsection;
- Add one glossary section;

- Clarify the relation between data object, object tag, property tag and metric tag in Self Describing Data Object Tags Use Case section;

- Add update to RFC8407 in the front page.

Authors' Addresses

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

Email: bill.wu@huawei.com

Benoit Claise
Huawei
De Kleetlaan 6a b1
Diegem 1831
Belgium

Email: benoit.claise@huawei.com

Peng Liu
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: liupengyjy@chinamobile.com

Zongpeng Du
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: duzongpeng@chinamobile.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Network Working Group
Internet-Draft
Updates: 7950,8407,8525 (if approved)
Intended status: Standards Track
Expires: August 26, 2021

R. Wilton, Ed.
Cisco Systems, Inc.
R. Rahman, Ed.

B. Lengyel, Ed.
Ericsson
J. Clarke
Cisco Systems, Inc.
J. Sterne
Nokia
B. Claise
Huawei
K. D'Souza
AT&T

February 22, 2021

Updated YANG Module Revision Handling
draft-ietf-netmod-yang-module-versioning-02

Abstract

This document specifies a new YANG module update procedure that can document when non-backwards-compatible changes have occurred during the evolution of a YANG module. It extends the YANG import statement with an earliest revision filter to better represent inter-module dependencies. It provides help and guidelines for managing the lifecycle of YANG modules and individual schema nodes. It provides a mechanism, via the revision-label YANG extension, to specify a revision identifier for YANG modules. This document updates RFC 7950, RFC 8407 and RFC 8525.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 26, 2021.

Table of Contents

1. Introduction

This document defines a solution to the YANG module lifecycle problems described in [I-D.ietf-netmod-yang-versioning-reqs]. Complementary documents provide a complete solution to the YANG versioning requirements, with the overall relationship of the solution drafts described in [I-D.ietf-netmod-yang-solutions].

Specifically, this document recognises a need (within standards organizations, vendors, and the industry) to sometimes allow YANG modules to evolve with non-backwards-compatible changes, which could cause breakage to clients and importing YANG modules. Accepting that non-backwards-compatible changes do sometimes occur, it is important to have mechanisms to report where these changes occur, and to manage their effect on clients and the broader YANG ecosystem.

The document comprises five parts:

Refinements to the YANG 1.1 module revision update procedure, supported by new extension statements to indicate when a revision contains non-backwards-compatible changes, and an optional revision label.

A YANG extension statement allowing YANG module imports to specify an earliest module revision that may satisfy the import dependency.

Updates and augmentations to ietf-yang-library to include the revision label in the module descriptions, to report how "deprecated" and "obsolete" nodes are handled by a server, and to

clarify how module imports are resolved when multiple revisions could otherwise be chosen.

Considerations of how versioning applies to YANG instance data.

Guidelines for how the YANG module update rules defined in this document should be used, along with examples.

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

Open issues are tracked at <https://github.com/netmod-wg/yang-ver-dt/issues>.

1.1. Updates to YANG RFCs

This document updates [RFC7950] section 11. Section 3 describes modifications to YANG revision handling and update rules, and Section 4 describes a YANG extension statement to do import by derived revision.

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

This document updates [RFC8407] section 4.7. Section 7 provides guidelines on managing the lifecycle of YANG modules that may contain non-backwards-compatible changes and a branched revision history.

This document updates [RFC7950] section 5.2. Section 3.3 describes the use of a revision label in the name of a file containing a YANG module or submodule.

2. Terminology and Conventions

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

In addition, this document uses the terminology:

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

- Backwards-compatible (BC) change: A backwards-compatible change between two YANG module revisions, as defined in Section 3.1.1

- Non-backwards-compatible (NBC) change: A non-backwards-compatible change between two YANG module revisions, as defined in Section 3.1.3

- Valuespace: The valuespace of a leaf or leaf-list is the set of values that the schema node may have according to the type and constraint statements of the YANG model.

3. Refinements to YANG revision handling

[RFC7950] assumes, but does not explicitly state, that the revision history for a YANG module is strictly linear, i.e., it is prohibited to have two independent revisions of a YANG module that are both directly derived from the same parent revision.

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

A corollary to the above is that the relationship between two module revisions cannot be determined by comparing the module revision date alone, and the revision history, or revision label, must also be taken into consideration.

A module's name and revision date identifies a specific immutable definition of that module within its revision history. Hence, if a module includes submodules then to ensure that the module's content is uniquely defined, the module's "include" statements SHOULD use "revision-date" substatements to specify the exact revision date of each included submodule. When a module does not include its submodules by revision-date, the revision of submodules used cannot be derived from the including module. Mechanisms such as YANG packages [I-D.ietf-netmod-yang-packages], and YANG library [[RFC7895] [RFC8525], MAY be used to specify the exact submodule revisions used when the submodule revision date is not constrained by the "include" statement.

[RFC7950] section 11 requires that all updates to a YANG module are BC to the previous revision of the module. This document allows for more flexible evolution of YANG modules: NBC changes between module revisions are allowed and are documented using a new "nbc-changes" YANG extension statement in the module revision history.

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

3.1. Updating a YANG module with a new revision

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

Where pragmatic, updates to YANG modules SHOULD be backwards-compatible, following the definition in Section 3.1.1.

A new module revision MAY contain NBC changes, i.e., the semantics of an existing definition MAY be changed in an NBC way without requiring a new definition with a new identifier. A new module revision with NBC changes MUST include the "rev:nbc-changes" extension substatement to signal the potential for incompatibility to existing module users and readers.

3.1.1. Backwards-compatible rules for config data

A change between two module revisions is defined as being "backwards-compatible" if the change conforms to the module update rules specified in [RFC7950] section 11, updated by the following rules:

- A "status" "deprecated" statement MAY be added, or changed from "current" to "deprecated", but adding or changing "status" to "obsolete" is not a backwards-compatible change.

- Obsolete definitions MAY be removed from published modules, and are classified as backwards-compatible changes. In some circumstances it may be helpful to retain the obsolete definitions to ensure that their identifiers are not reused with a different meaning.

- In statements that have any data definition statements as substatements, those data definition substatements MAY be reordered, as long as they do not change the ordering of any "input" or "output" data definition substatements of "rpc" or "action" statements. If new data definition statements are added, they can be added anywhere in the sequence of existing substatements.

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

3.1.2. Backwards-compatibility rules for config false and output data

Compatibility behavior of configuration and state data is different. While adding a mandatory configuration leaf makes earlier configurations invalid, an additional state leaf can easily be discarded. Decreasing the valuespace of a configuration leaf makes any configuration invalid that uses the newly excluded values; decreasing the valuespace of a state schema node should not disturb a client application. Data in the output section of notifications, actions or rpcs is governed by the same backwards compatibility behavior as config=false data.

While incoming configuration data is checked according to YANG constraints, constraints on state data sent by the server MAY or MAY NOT be enforced. The following guidelines are provided for client application designers to allow a smooth interworking with servers.

- A client MUST tolerate any data received (or not received) without crashing.
- A client MUST be able to discard any data that is not part of the model but is sent by the server additionally (e.g. XML elements or attributes, JSON properties).
- A client SHOULD be able to handle valid parts of a received data set even if it discards other parts as invalid.
- A client SHOULD be able to handle data that is outside the valuespace defined, as long as it is of the same basic type.
- A client SHOULD be prepared to handle more items for a list or leaf-list than what is defined by the model.

Based on the above client guidelines and the intent to allow the correct and flexible handling of state and output data even after module revision changes the following rules define which config false and output schema changes are considered BC or NBC. The rules reflect common client behavior, however a client that expects a specific server behavior or data set may have problems with any change. The rules are defined as a compromise between protecting client applications and allowing the most common changes.

- Adding mandatory or optional schema nodes is BC
- Changing an optional schema node to mandatory is BC
- Removal of a mandatory or optional schema node is NBC

- Changing a mandatory schema node to optional is NBC
- Expanding the valuespace of a leaf or leaf-list is BC. Change of the valuespace may be the result of a change to a range, length, pattern, base, enum, bit, require-instance or must statements.
- Decreasing the valuespace of a leaf or leaf-list is BC
- Changing max-elements is BC
- Increasing min-element is BC
- Changing min-elements to a lower value is NBC (it is like removing mandatory)
- Modifying the type of a leaf or leaf-list is NBC

3.1.3. Non-backwards-compatible changes

Any changes to YANG modules that are not defined by Section 3.1.1 or Section 3.1.2 as being backwards-compatible are classified as "non-backwards-compatible" changes.

3.2. nbc-changes revision extension statement

The "rev:nbc-changes" extension statement is used to indicate YANG module revisions that contain NBC changes.

If a revision of a YANG module contains changes, relative to the preceding revision in the revision history, that do not conform to the module update rules defined in Section 3.1.1 or Section 3.1.2, then a "rev:nbc-changes" extension statement MUST be added as a substatement to the "revision" statement.

Conversely, if a revision does not contain an "rev:nbc-changes" extension substatement then all changes, relative to the preceding revision in the revision history, MUST be backwards-compatible.

3.3. Revision label

This section updates [RFC7950] section 5.2, it explains how a revision label can be used in the name of a file containing a YANG module or submodule.

Each revision entry in a module or submodule MAY have a revision label associated with it, providing an alternative alias to identify a particular revision of a module or submodule. The revision label

could be used to provide an additional versioning identifier associated with the revision.

YANG Semver [I-D.ietf-netmod-yang-semver] defines a versioning scheme based on Semver 2.0.0 [semver] that can be used as a revision label.

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

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

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

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

If a revision has an associated revision label, then the revision-label may be used instead of the revision date in the filename of a YANG file, where it takes the form:

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

    E.g., acme-router-module@2018-01-25.yang
    E.g., acme-router-module#2.0.3.yang
```

YANG module (or submodule) files MAY be identified using either revision-date or revision-label. Typically, only one file name SHOULD exist for the same module (or submodule) revision. Two file names, one with the revision date and another with the revision label, MAY exist for the same module (or submodule) revision, e.g when migrating from one scheme to the other.

3.3.1. Revision label scheme extension statement

The "rev:revision-label-scheme" extension statement is used to indicate which revision-label scheme a module or submodule uses. The mandatory argument to this extension statement:

- Specifies the revision-label scheme used by the module or submodule
- Is defined in the document which specifies the revision-label scheme
- MUST be an identity derived from "revision-label-scheme-base"

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

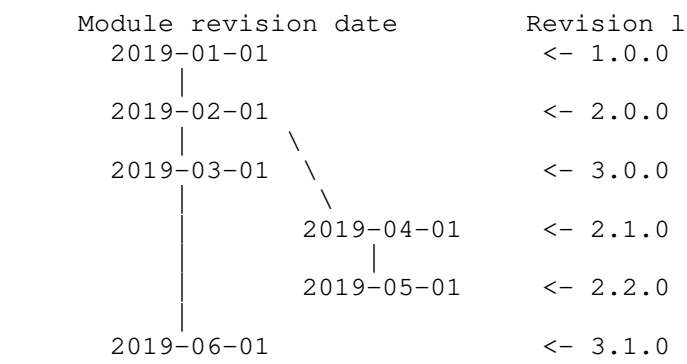
3.3.2. Removing revisions from the revision history

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

3.4. Examples for updating the YANG module revision history

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

Example YANG module with branched revision history.



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

Example module, revision 2019-06-01:

```
    module example-module {

      namespace "urn:example:module";
      prefix "prefix-name";

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

      description
        "to be completed";

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

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

      revision 2019-02-01 {
        rev:revision-label 2.0.0;
        rev:nbc-changes;
        description "Apply bugfix to pattern statement";
      }

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

      //YANG module definition starts here
    }
```

Example module, revision 2019-05-01:

```
    module example-module {

      namespace "urn:example:module";
      prefix "prefix-name";

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

      description
        "to be completed";

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

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

      revision 2019-02-01 {
        rev:revision-label 2.0.0;
        rev:nbc-changes;
        description "Apply bugfix to pattern statement";
      }

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

      //YANG module definition starts here
    }
```

4. Import by derived revision

RFC 7950 allows YANG module "import" statements to optionally require the imported module to have a particular revision date. In practice, importing a module with an exact revision date is often too restrictive because it requires the importing module to be updated whenever any change to the imported module occurs. The alternative choice of using an import statement without any revision date statement is also not ideal because the importing module may not work with all possible revisions of the imported module.

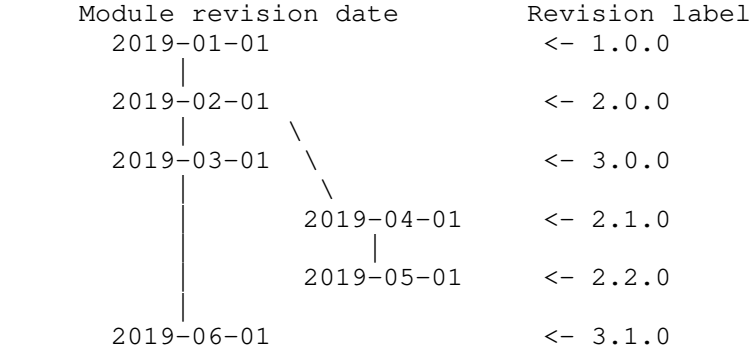
Instead, it is desirable for a importing module to specify a "minimum required revision" of a module that it is compatible with, based on the assumption that later revisions derived from that "minimum required revision" are also likely to be compatible. Many possible changes to a YANG module do not break importing modules, even if the changes themselves are not strictly backwards-compatible. E.g., fixing an incorrect pattern statement or description for a leaf would not break an import, changing the name of a leaf could break an import but frequently would not, but removing a container would break imports if that container is augmented by another module.

The ietf-revisions module defines the "revision-or-derived" extension statement, a substatement to the YANG "import" statement, to allow for a "minimum required revision" to be specified during import:

> The argument to the "revision-or-derived" extension statement is a revision date or a revision label.
>
> A particular revision of an imported module satisfies an import's "revision-or-derived" extension statement if the imported module's revision history contains a revision statement with a matching revision date or revision label.
>
> An "import" statement MUST NOT contain both a "revision-or-derived" extension statement and a "revision-date" statement.
>
> The "revision-or-derived" extension statement MAY be specified multiple times, allowing the import to use any module revision that satisfies at least one of the "revision-or-derived" extension statements.
>
> The "revision-or-derived" extension statement does not guarantee that all module revisions that satisfy an import statement are necessarily compatible, it only gives an indication that the revisions are more likely to be compatible. Hence, NBC changes to an imported module may also require new revisions of any importing modules, updated to accommodation those changes, along with updated import "revision-or-derived" extension statements to depend on the updated imported module revision.
>
> Adding, modifying or removing a "revision-or-derived" extension statement is considered to be a BC change.
>
> Adding, modifying or removing a "revision-date" extension statement is considered to be a BC change.

4.1. Module import examples

Consider the example module "example-module" from Section 3.4 that is hypothetically available in the following revision/label pairings: 2019-01-01/1.0.0, 2019-02-01/2.0.0, 2019-03-01/3.0.0, 2019-04-01/2.1.0, 2019-05-01/2.2.0 and 2019-06-01/3.1.0. The relationship between the revisions is as before:



4.1.1. Example 1

This example selects module revisions that match, or are derived from the revision 2019-02-01. E.g., this dependency might be used if there was a new container added in revision 2019-02-01 that is augmented by the importing module.It includes revisions/labels: 2019-02-01/2.0.0, 2019-03-01/3.0.0, 2019-04-01/2.1.0, 2019-05-01/2.2.0 and 2019-06-01/3.1.0.

```
import example-module {
  rev:revision-or-derived 2019-02-01;
}
```

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

```
import example-module {
  rev:revision-or-derived 2.0.0;
}
```

4.1.2. Example 2

This example selects module revisions that are derived from 2019-04-01 by using the revision label 2.1.0. It includes revisions/labels: 2019-04-01/2.1.0 and 2019-05-01/2.2.0. Even though 2019-06-01/3.1.0 has a higher revision label number than

2019-04-01/2.1.0 it is not a derived revision, and hence it is not a valid revision for import.

```
import example-module {
  rev:revision-or-derived 2.1.0;
}
```

4.1.3. Example 3

This example selects revisions derived from either 2019-04-01 or 2019-06-01. It includes revisions/labels: 2019-04-01/2.1.0, 2019-05-01/2.2.0, and 2019-06-01/3.1.0.

```
import example-module {
  rev:revision-or-derived 2019-04-01;
  rev:revision-or-derived 2019-06-01;
}
```

5. Updates to ietf-yang-library

This document updates YANG library [RFC7950] to clarify how ambiguous module imports are resolved. It also defines the YANG module, ietf-yang-library-revisions that augments YANG library [RFC8525] with new revision-label related meta-data.

5.1. Resolving ambiguous module imports

A YANG datastore schema, defined in [RFC8525], can specify multiple revisions of a YANG module in the schema using the "import-only" list, with the requirement from [RFC7950] that only a single revision of a YANG module may be implemented.

If a YANG module import statement does not specify a specific revision within the datastore schema then it could be ambiguous as to which module revision the import statement should resolve to. Hence, a datastore schema constructed by a client using the information contained in YANG library may not exactly match the datastore schema actually used by the server.

The following two rules remove the ambiguity:

If a module import statement could resolve to more than one module revision defined in the datastore schema, and one of those revisions is implemented (i.e., not an "import-only" module), then the import statement MUST resolve to the revision of the module that is defined as being implemented by the datastore schema.

If a module import statement could resolve to more than one module revision defined in the datastore schema, and none of those revisions are implemented, then the import MUST resolve to the module revision with the latest revision date.

5.2. YANG library versioning augmentations

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

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

5.2.1. Advertising revision-label

The ietf-yang-library-revisions YANG module augments the "module" list in ietf-yang-library with a "revision-label" leaf to optionally declare the revision label associated wth the particular revision of each module.

5.2.2. Reporting how deprecated and obsolete nodes are handled

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

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

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

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

If a server does not set the "deprecated-nodes-implemented" leaf to "true", then clients MUST NOT rely solely on the "rev:nbc-changes" statements to determine whether two module revisions are backwards-compatible, and MUST also consider whether the status of any nodes has changed to "deprecated" and whether those nodes are implemented by the server.

6. Versioning of YANG instance data

Instance data sets [I-D.ietf-netmod-yang-instance-file-format] do not directly make use of the updated revision handling rules described in this document, as compatibility for instance data is undefined.

However, instance data specifies the content-schema of the data-set. This schema SHOULD make use of versioning using revision dates and/or revision labels for the individual YANG modules that comprise the schema or potentially for the entire schema itself (e.g., [I-D.ietf-netmod-yang-packages]).

In this way, the versioning of a content-schema associated with an instance data set may help a client to determine whether the instance data could also be used in conjunction with other revisions of the YANG schema, or other revisions of the modules that define the schema.

7. Guidelines for using the YANG module update rules

The following text updates section 4.7 of [RFC8407] to revise the guidelines for updating YANG modules.

7.1. Guidelines for YANG module authors

All IETF YANG modules MUST include revision-label statements for all newly published YANG modules, and all newly published revisions of existing YANG modules. The revision-label MUST take the form of a YANG semantic version number [I-D.ietf-netmod-yang-semver].

NBC changes to YANG modules may cause problems to clients, who are consumers of YANG models, and hence YANG module authors are RECOMMENDED to minimize NBC changes and keep changes BC whenever possible.

When NBC changes are introduced, consideration should be given to the impact on clients and YANG module authors SHOULD try to mitigate that impact.

A "rev:nbc-changes" statement MUST be added if there are NBC changes relative to the previous revision.

Removing old revision statements from a module's revision history could break import by revision, and hence it is RECOMMENDED to retain them. If all depencencies have been updated to not import specific revisions of a module, then the corresponding revision statements can be removed from that module. An alternative solution, if the revision section is too long, would be remove, or curtail, the older description statements associated with the previous revisions.

The "rev:revision-or-derived" extension should be used in YANG module imports to indicate revision dependencies between modules in preference to the "revision-date" statement, which causes overly strict import dependencies and SHOULD NOT be used.

A module that includes submodules SHOULD use the "revision-date" statement to include specific submodule revisions. The revision of the including module MUST be updated when any included submodule has changed. The revision-label substatement used in the new module revision MUST indicate the nature of the change, i.e. NBC or BC, to the module's schema tree.

7.1.1. Making non-backwards-compatible changes to a YANG module

There are various valid situations where a YANG module has to be modified in an NBC way. Here are the different ways in which this can be done:

- NBC changes can be sometimes be done incrementally using the "deprecated" status to provide clients time to adapt to NBC changes.
- NBC changes are done at once, i.e. without using "status" statements. Depending on the change, this may have a big impact on clients.
- If the server can support multiple revisions of the YANG module or of YANG packages(as specified in [I-D.ietf-netmod-yang-packages]), and allows the client to select the revision (as per [I-D.ietf-netmod-yang-ver-selection]), then NBC changes MAY be done without using "status" statements. Clients would be required to select the revision which they support and the NBC change would have no impact on them

Here are some guidelines on how non-backwards-compatible changes can be made incrementally, with the assumption that deprecated nodes are implemented by the server, and obsolete nodes are not:

1. The changes should be made gradually, e.g. a data node's status SHOULD NOT be changed directly from "current" to "obsolete" (see

Section 4.7 of [RFC8407]), instead the status SHOULD first be marked "deprecated" and then when support is removed its status MUST be changed to "obsolete". Instead of using the "obsolete" status, the data node MAY be removed from the model but this has the risk of breaking modules which import the modified module.

2. For deprecated data nodes the "description" statement SHOULD also indicate until when support for the node is guaranteed (if known). If there is a replacement data node, rpc, action or notification for the deprecated node, this SHOULD be stated in the "description". The reason for deprecating the node can also be included in the "description" if it is deemed to be of potential interest to the user.

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

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

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

See Appendix B for examples on how NBC changes can be made.

7.2. Versioning Considerations for Clients

Guidelines for clients of modules using the new module revision update procedure:

- Clients SHOULD be liberal when processing data received from a server. For example, the server may have increased the range of an operational node causing the client to receive a value which is outside the range of the YANG model revision it was coded against.

- Clients SHOULD monitor changes to published YANG modules through their revision history, and use appropriate tooling to understand the specific changes between module revision. In particular, clients SHOULD NOT migrate to NBC revisions of a module without understanding any potential impact of the specific NBC changes.

- Clients SHOULD plan to make changes to match published status changes. When a node's status changes from "current" to "deprecated", clients SHOULD plan to stop using that node in a timely fashion. When a node's status changes to "obsolete", clients MUST stop using that node.

8. Module Versioning Extension YANG Modules

YANG module with extension statements for annotating NBC changes, revision label, revision label scheme, and importing by revision.

```
<CODE BEGINS> file "ietf-yang-revisions@2021-02-17.yang"
 module ietf-yang-revisions {
   yang-version 1.1;
   namespace "urn:ietf:params:xml:ns:yang:ietf-yang-revisions";
   prefix rev;

   // RFC Ed.: We need the bis version to get the new type revision-identifier
   // If 6991-bis is not yet an RFC we need to copy the definition here
   import ietf-yang-types {
     prefix yang;
     reference
       "XXXX [ietf-netmod-rfc6991-bis]: Common YANG Data Types";
   }

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

      Author:   Benoit Claise
                <mailto:benoit.claise@huawei.com>

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

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

      Author:   Robert Wilton
                <mailto:rwilton@cisco.com>
```

```
      Author:   Kevin D'Souza
                <mailto:kd6913@att.com>

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

      Author:   Jason Sterne
                <mailto:jason.sterne@nokia.com>";
   description
     "This YANG 1.1 module contains definitions and extensions to
      support updated YANG revision handling.

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

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

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

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

   // RFC Ed.: update the date below with the date of RFC publication
   // and remove this note.
   // RFC Ed.: replace XXXX (inc above) with actual RFC number and
   // remove this note.

   revision 2021-02-17 {
     description
       "Initial version.";
     reference
       "XXXX: Updated YANG Module Revision Handling";
   }

   typedef revision-label {
     type string {
       length "1..255";
       pattern '[a-zA-Z0-9,\-_.+]+';
       pattern '\d{4}-\d{2}-\d{2}' {
```

```
        modifier invert-match;
      }
    }
    description
      "A label associated with a YANG revision.

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

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

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

       The statement MUST only be a substatement of the 'revision'
       statement.
       Zero or one 'nbc-changes' statement per parent statement is
       allowed.
       The statement MUST NOT have any substatements.

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

       Conversely, if a revision of a YANG module only contains
       changes, relative to the preceding revision in the revision
       history, that are classified as 'backwards-compatible' then
       the revision statement MUST NOT contain any 'nbc-changes'
       substatement.";

    reference
      "XXXX: Updated YANG Module Revision Handling;
```

```
        Section 3.2, nbc-changes revision extension statement";
    }

    extension revision-label {
      argument revision-label;
      description
        "The revision label can be used to provide an additional
         versioning identifier associated with the revision.  E.g., one
         option for a versioning scheme that could be used is [TODO -
         Reference semver draft].

         The format of the revision-label argument MUST conform to the
         pattern defined for the revision-label typedef.

         The statement MUST only be a substatement of the revision
         statement.
         Zero or one revision-label statement per parent statement
         is allowed.
         The statement MUST NOT have any substatements.

         Revision labels MUST be unique amongst all revisions of a
         module.";

      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 3.3, Revision label";
    }

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

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

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

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

         The statement MUST NOT have any substatements.";

      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 3.3.1, Revision label scheme extension statement";
```

```
   }

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

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

        The statement MUST only be a substatement of the import
        statement.
        Zero, one or more 'revision-or-derived' statement per parent
        statement is allowed.
        The statement MUST NOT have any substatements.

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

        An 'import' statement MUST NOT contain both a
        'revision-or-derived' extension statement and a
        'revision-date' statement.

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

        The 'revision-or-derived' extension statement does not
        guarantee that all module revisions that satisfy an import
        statement are necessarily compatible, it only gives an
        indication that the revisions are more likely to be
        compatible.";

     reference
       "XXXX: Updated YANG Module Revision Handling;
        Section 4, Import by derived revision";
   }

   identity revision-label-scheme-base {
     description
       "Base identity from which all revision label schemes are
        derived.";
```

```
       reference
         "XXXX: Updated YANG Module Revision Handling;
          Section 3.3.1, Revision label scheme extension statement";

   }
 }
<CODE ENDS>
```

YANG module with augmentations to YANG Library to revision labels

```
<CODE BEGINS> file "ietf-yang-library-revisions@2020-07-06.yang"
module ietf-yang-library-revisions {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-library-revisions";
  prefix yl-rev;

  import ietf-yang-revisions {
    prefix rev;
    reference
      "XXXX: Updated YANG Module Revision Handling";
  }

  import ietf-yang-library {
    prefix yanglib;
    reference "RFC 8525: YANG Library";
  }

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

     Author:   Benoit Claise
               <mailto:benoit.claise@huawei.com>

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

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

     Author:   Robert Wilton
               <mailto:rwilton@cisco.com>

     Author:   Kevin D'Souza
               <mailto:kd6913@att.com>
```

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

     Author:   Jason Sterne
               <mailto:jason.sterne@nokia.com>";
  description
    "This module contains augmentations to YANG Library to add module
     level revision label and to provide an indication of how
     deprecated and obsolete nodes are handled by the server.

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

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

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

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

  // RFC Ed.: update the date below with the date of RFC publication
  // and remove this note.
  // RFC Ed.: replace XXXX (including in the imports above) with
  // actual RFC number and remove this note.
  // RFC Ed.: please replace revision-label version with 1.0.0 and
  // remove this note.
  revision 2020-07-06 {
    rev:revision-label 0.1.0;
    description
      "Initial revision";
    reference
      "XXXX: Updated YANG Module Revision Handling";
  }

  augment "/yanglib:yang-library/yanglib:module-set/yanglib:module" {
    description
      "Augmentation modules with a revision label";
    leaf revision-label {
      type rev:revision-label;
```

```
      description
        "The revision label associated with this module revision.
         The label MUST match the rev:label value in the specific
         revision of the module loaded in this module-set.";

      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 5.2.1, Advertising revision-label";
    }
  }

  augment "/yanglib:yang-library/yanglib:schema" {
    description
      "Augmentations to the ietf-yang-library module to indicate how
       deprecated and obsoleted nodes are handled for each datastore
       schema supported by the server.";

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

      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 5.2.2, Reporting how deprecated and obsolete nodes
         are handled";
    }

    leaf obsolete-nodes-absent {
      type boolean;
      description
        "If set to true, this leaf indicates that the server does not
         implement any status 'obsolete' nodes.  If this leaf is
         set to false or absent, then the behaviour is unspecified.";

      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 5.2.2, Reporting how deprecated and obsolete nodes
         are handled";
    }
  }
}
<CODE ENDS>
```

9. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The following individuals are (or have been) members of the design team and have worked on the YANG versioning project:

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

The initial revision of this document was refactored and built upon [I-D.clacla-netmod-yang-model-update].

Discussons on the use of Semver for YANG versioning has been held with authors of the OpenConfig YANG models. We would like thank both Anees Shaikh and Rob Shakir for their input into this problem space.

We would also like to thank Martin Bjorklund, Jan Lindblad and Italo Busi for their contributions.

10. Security Considerations

The document does not define any new protocol or data model. There are no security considerations beyond those specified in [RFC7950].

11. IANA Considerations

11.1. YANG Module Registrations

The following YANG module is requested to be registred in the "IANA Module Names" registry:

The ietf-yang-revisions module:

Name: ietf-yang-revisions

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-revisions

Prefix: rev

Reference: [RFCXXXX]

The ietf-yang-library-revisions module:

Name: ietf-yang-library-revisions

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-library-revisions

Prefix: yl-rev

Reference: [RFCXXXX]

11.2. Guidance for versioning in IANA maintained YANG modules

Note for IANA (to be removed by the RFC editor): Please check that the registries and IANA YANG modules are referenced in the appropriate way.

IANA is responsible for maintaining and versioning YANG modules that are derived from other IANA registries. For example, "iana-if-type.yang" [IfTypeYang] is derived from the "Interface Types (ifType) IANA registry" [IfTypesReg], and "iana-routing-types.yang" [RoutingTypesYang] is derived from the "Address Family Numbers" [AddrFamilyReg] and "Subsequent Address Family Identifiers (SAFI) Parameters" [SAFIReg] IANA registries.

Normally, updates to the registries cause any derived YANG modules to be updated in a backwards-compatible way, but there are some cases where the registry updates can cause non-backward-compatible updates to the derived YANG module. An example of such an update is the 2020-12-31 revision of iana-routing-types.yang

[RoutingTypesDecRevision], where the enum name for two SAFI values was changed.

In all cases, IANA MUST follow the versioning guidance specified in Section 3.1, and MUST include a "rev:nbc-changes" substatement to the latest revision statement whenever an IANA maintained module is updated in a non-backwards-compatible way, as described in Section 3.2.

Note: For published IANA maintained YANG modules that contain non-backwards-compatible changes between revisions, a new revision should be published with the "rev:nbc-changes" substatement retrospectively added to any revisions containing non-backwards-compatible changes.

Non normative examples of updates to enumeration types in IANA maintained modules that would be classified as non-backwards-compatible changes are: Changing the status of an enumeration typedef to obsolete, changing the status of an enum entry to obsolete, removing an enum entry, changing the identifier of an enum entry, or changing the described meaning of an enum entry.

Non normative examples of updates to enumeration types in IANA maintained modules that would be classified as backwards-compatible changes are: Adding a new enum entry to the end of the enumeration, changing the status or an enum entry to deprecated, or improving the description of an enumeration that does not change its defined meaning.

Non normative examples of updates to identity types in IANA maintained modules that would be classified as non-backwards-compatible changes are: Changing the status of an identity to obsolete, removing an identity, renaming an identity, or changing the described meaning of an identity.

Non normative examples of updates to identity types in IANA maintained modules that would be classified as backwards-compatible changes are: Adding a new identity, changing the status or an identity to deprecated, or improving the description of an identity that does not change its defined meaning.

12. References

12.1. Normative References

[I-D.ietf-netmod-rfc6991-bis]
Schoenwaelder, J., "Common YANG Data Types", draft-ietf-netmod-rfc6991-bis-04 (work in progress), July 2020.

[I-D.ietf-netmod-yang-semver]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "YANG Semantic Versioning", draft-ietf-netmod-yang-semver-01 (work in progress), July 2020.

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

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <https://www.rfc-editor.org/info/rfc7895>.

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

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

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

12.2. Informative References

[AddrFamilyReg]
"Address Family Numbers IANA Registry", <https://www.iana.org/assignments/address-family-numbers/address-family-numbers.xhtml>.

[I-D.clacla-netmod-yang-model-update]
Claise, B., Clarke, J., Lengyel, B., and K. D'Souza, "New YANG Module Update Procedure", draft-clacla-netmod-yang-model-update-06 (work in progress), July 2018.

[I-D.ietf-netmod-yang-instance-file-format]
Lengyel, B. and B. Claise, "YANG Instance Data File Format", draft-ietf-netmod-yang-instance-file-format-12 (work in progress), April 2020.

[I-D.ietf-netmod-yang-packages]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and W. Bo, "YANG Packages", draft-ietf-netmod-yang-packages-01 (work in progress), November 2020.

[I-D.ietf-netmod-yang-solutions]
Wilton, R., "YANG Versioning Solution Overview", draft-ietf-netmod-yang-solutions-01 (work in progress), November 2020.

[I-D.ietf-netmod-yang-ver-selection]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and W. Bo, "YANG Schema Selection", draft-ietf-netmod-yang-ver-selection-00 (work in progress), March 2020.

[I-D.ietf-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", draft-ietf-netmod-yang-versioning-reqs-04 (work in progress), January 2021.

[IfTypesReg]
"Interface Types (ifType) IANA Registry", <https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-5>.

[IfTypeYang]
"iana-if-type YANG Module", <https://www.iana.org/assignments/iana-if-type/iana-if-type.xhtml>.

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

[RoutingTypesDecRevision]
"2020-12-31 revision of iana-routing-types.yang", <https://www.iana.org/assignments/yang-parameters/iana-routing-types@2020-12-31.yang>.

[RoutingTypesYang]
"iana-routing-types YANG Module", <https://www.iana.org/assignments/iana-routing-types/iana-routing-types.xhtml>.

[SAFIReg] "Subsequent Address Family Identifiers (SAFI) Parameters IANA Registry", <https://www.iana.org/assignments/safi-namespace/safi-namespace.xhtml>.

[semver] "Semantic Versioning 2.0.0", <https://www.semver.org>.

Appendix A. Examples of changes that are NBC

Examples of NBC changes include:

- Deleting a data node, or changing it to status obsolete.
- Changing the name, type, or units of a data node.
- Modifying the description in a way that changes the semantic meaning of the data node.
- Any changes that change or reduce the allowed value set of the data node, either through changes in the type definition, or the addition or changes to "must" statements, or changes in the description.
- Adding or modifying "when" statements that reduce when the data node is available in the schema.
- Making the statement conditional on if-feature.

Appendix B. Examples of applying the NBC change guidelines

The following sections give guidance for how some of these NBC changes could be made to a YANG module. The examples are all for "config true" nodes.

B.1. Removing a data node

Removing a leaf or container from the data tree, e.g. because support for the corresponding feature is being removed:

1. The node's status is changed to "deprecated" and it is supported for at least one year. This is a BC change.
2. When the node is not available anymore, its status is changed to "obsolete" and the "description" updated, this is an NBC change.

If the server can support NBC revisions of the YANG module simultaneously using version selection [I-D.ietf-netmod-yang-ver-selection], then the changes can be done immediately:

1. The new revision of the YANG module has the node's status changed to "obsolete" and the "description" updated, this is an NBC change.

2. Clients which require the data node select the YANG package containing the schema version they use

B.2. Changing the type of a leaf node

Changing the type of a leaf-node. e.g. consider a "vpn-id" node of type integer being changed to a string:

1. The status of node "vpn-id" is changed to "deprecated" and the node should be available for at least one year. This is a BC change.

2. A new node, e.g. "vpn-name", of type string is added to the same location as the existing node "vpn-id". This new node has status "current" and its description explains that it is replacing node "vpn-id".

3. During the period of time where both nodes are available, how the server behaves when either node is set is outside the scope of this document and will vary on a case by case basis. Here are some options:

    1. A server may prevent the new node from being set if the old node is already set (and vice-versa). The new node may have a when statement to achieve this. The old node must not have a when statement since this would be an NBC change, but the server could reject the old node from being set if the new node is already set.

    2. If the new node is set and a client does a get or get-config operation on the old node, the server could map the value. For example, if the new node "vpn-name" has value "123" then the server could return integer value 123 for the old node "vpn-id". However, if the value can not be mapped then the configuration would be incomplete, this is outside the scope of this document.

4. When node "vpn-id" is not available anymore, its status is changed to "obsolete" and the "description" is updated. This is an NBC change.

If the server can support NBC revisions of the YANG module simultaneously using version selection

[I-D.ietf-netmod-yang-ver-selection], then the changes can be done immediately:

1. In the new revision of the YANG module, the status of node "vpn-id" is changed to "obsolete". This is an NBC change.

2. New node "vpn-name" is added to the same location as described above.

3. Clients which require the data node select the YANG package containing the schema version they use

4. A server should not map between the nodes "vpn-id" and "vpn-name", i.e. if a client creates a data instance with "vpn-name" then that data instance should not be visible to a client using a module revision which has "vpn-id" (and vice-versa).

B.3. Reducing the range of a leaf node

Reducing the range of values of a leaf-node. e.g. consider a "vpn-id" node of type integer being changed from type uint32 to type uint16:

1. If all values which are being removed were never supported, e.g. if a vpn-id of 65536 or higher was never accepted, this is a BC change for the functionality (no functionality change). Even if it is an NBC change for the YANG model, there should be no impact for clients using that YANG model.

2. If one or more values being removed was previously supported, e.g. if a vpn-id of 65536 was accepted previously, this is an NBC change for the YANG model. Clients using the old YANG model will be impacted, so a change of this nature should be done carefully, e.g. by using the steps described in Appendix B.2

B.4. Changing the key of a list

Changing the key of a list has a big impact to the client. For example, consider a "sessions" list which has a key "interface" and there is a need to change the key to "dest-address", such a change can be done in steps:

1. The status of list "sessions" is changed to "deprecated" and the list should be available for at least one year. This is a BC change.

2. A new list is created in the same location with the same data but with "dest-address" as key. Finding an appropriate name for the new list can be tricky especially if the name of the existing

list was perfect. In this case the new list is called "sessions-address", has status "current" and its description should explain that it is replacing list "session".

3. During the period of time where both lists are available, how the server behaves when either list is set is outside the scope of this document and will vary on a case by case basis. Here are some options:

    1. A server could prevent the new list from being set if the old list already has entries (and vice-versa).

    2. If the new list is set and a client does a get or get-config operation on the old list, the server could map the entries. However if the new list has entries which would lead to duplicate keys in the old list, the mapping can not be done.

4. When list "sessions" is not available anymore, its status is changed to "obsolete" and the "description" is updated. This is an NBC change.

If the server can support NBC revisions of the YANG module simultaneously using version selection [I-D.ietf-netmod-yang-ver-selection], then the changes can be done immediately:

1. The new revision of the YANG module has the list "sessions" modified to have "dest-address" as key, this is an NBC change.

2. Clients which require the previous functionality select the older module revision

B.5. Renaming a node

A leaf-node or a container may be renamed, either due to a spelling error in the previous name or because of a better name. For example a node "ip-adress" could be renamed to "ip-address":

1. The status of the existing node "ip-adress" is changed to "deprecated" and the node should be available for at least one year. This is a BC change.

2. The new node "ip-address" is added to the same location as the existing node "ip-adress". This new node has status "current" and its description should explain that it is replacing node "ip-adress".

3. During the period of time where both nodes are available, how the server behaves when either node is set is outside the scope of this document and will vary on a case by case basis. Here are some options:

    1. A server could prevent the new node from being set if the old node is already set (and vice-versa). The new node could have a when statement to achieve this. The old node must not have a when statement since this would be an NBC change, but the server could reject the old node from being set if the new node is already set.

    2. If the new node is set and a client does a get or get-config operation on the old node, the server could use the value of the new node. For example, if the new node "ip-address" has value X then the server may return value X for the old node "ip-adress".

4. When node "ip-adress" is not available anymore, its status is changed to "obsolete" and the "description" is updated. This is an NBC change.

If the server can support NBC revisions of the YANG module simultaneously using version selection [I-D.ietf-netmod-yang-ver-selection], then the changes can be done immediately:

1. The new revision of the YANG module has the node with the new name replacing the node with the old name, this is an NBC change.

2. Clients which require the previous node name select the older module revision

B.6. Changing a default value

Appendix C. Changes between revisions

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

v00 - v01

- Removed status-description

- Allowed both revision-date and revision-label in the filename.

- New extension revision-label-scheme

- To include submodules, inclusion by revision-date changed from MUST to SHOULD
- Submodules can use revision label scheme and it can be same or different as the including module's scheme
- Addressed various comments provided at WG adoption on rev-00

Authors' Addresses

Robert Wilton (editor)
Cisco Systems, Inc.

Email: rwilton@cisco.com

Reshad Rahman (editor)

Email: reshad@yahoo.com

Balazs Lengyel (editor)
Ericsson

Email: balazs.lengyel@ericsson.com

Joe Clarke
Cisco Systems, Inc.

Email: jclarke@cisco.com

Jason Sterne
Nokia

Email: jason.sterne@nokia.com

Benoit Claise
Huawei

Email: benoit.claise@huawei.com

Kevin D'Souza
AT&T

Email: kd6913@att.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2021

R. Wilton, Ed.
R. Rahman
J. Clarke
Cisco Systems, Inc.
J. Sterne
Nokia
B. Wu, Ed.
Huawei

November 2, 2020

YANG Packages
draft-ietf-netmod-yang-packages-01

Abstract

This document defines YANG packages, a versioned organizational structure holding a set of related YANG modules that collectively define a YANG schema. It describes how packages: are represented on a server, can be defined in offline YANG instance data files, and can be used to define the schema associated with YANG instance data files.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 6, 2021.

Table of Contents

1. Terminology and Conventions

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

This document uses terminology introduced in the YANG versioning requirements draft [I-D.ietf-netmod-yang-versioning-reqs].

This document also makes of the following terminology introduced in the Network Management Datastore Architecture [RFC8342]:

- datastore schema

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

- data node

In addition, this document defines the following terminology:

- YANG schema: A datastore schema, not bound to any particular datastore.

- YANG package: An organizational structure containing a collection of YANG modules, normally defined in a YANG instance data file. A YANG package defines a YANG schema by specifying a set of YANG modules and their revisions, other packages and their revisions, mandatory features, and deviations. YANG packages are defined in Section 5.

- backwards-compatible (BC) change: When used in the context of a YANG module, it follows the definition in Section 3.1.1 of [I-D.ietf-netmod-yang-module-versioning]. When used in the

context of a YANG package, it follows the definition in Section 5.2.1.2.

- non-backwards-compatible (NBC) change: When used in the context of a YANG module, it follows the definition in Section 3.1.2 of [I-D.ietf-netmod-yang-module-versioning]. When used in the context of a YANG package, it follows the definition in Section 5.2.1.2.

- editorial change: When used in the context of a YANG module, it follows the definition of an 'editorial change' in 3.2 of [I-D.ietf-netmod-yang-module-versioning]. When used in the context of a YANG package, it follows the definition in Section 5.2.1.3.

2. Introduction

This document defines and describes the YANG [RFC7950] constructs that are used to define and use YANG packages.

A YANG package is an organizational structure that groups a set of YANG modules together into a consistent versioned definition. For example, a YANG package could define the set of YANG modules required to implement an L2VPN service on a network device. YANG packages can themselves refer to, and reuse, other package definitions.

Non-normative examples of YANG packages are provided in the appendices.

3. Background on YANG packages

It has long been acknowledged within the YANG community that network management using YANG requires a unit of organization and conformance that is broader in scope than individual YANG modules.

'The YANG Package Statement' [I-D.bierman-netmod-yang-package] proposed a YANG package mechanism based on new YANG language statements, where a YANG package is defined in a file similar to how YANG modules are defined, and would require enhancements to YANG compilers to understand the new statements used to define packages.

OpenConfig [openconfigsemver] describes an approach to versioning 'bundle releases' based on git tags. I.e. a set of modules, at particular versions, can be marked with the same release tag to indicate that they are known to interoperate together.

The NETMOD WG in general, and the YANG versioning design team in particular, are exploring solutions [I-D.ietf-netmod-yang-solutions]

to the YANG versioning requirements, [I-D.ietf-netmod-yang-versioning-reqs]. Solutions to the versioning requirements can be split into several distinct areas. [I-D.ietf-netmod-yang-module-versioning] is focused on YANG versioning scoped to individual modules. The overall solution must also consider YANG versioning and conformance scoped to YANG schema. YANG packages provide part of the solution for versioning YANG schema.

4. Objectives

The main goals of YANG package definitions include, but are not restricted to:

- To provide an alternative, simplified, YANG conformance mechanism. Rather than conformance being performed against a set of individual YANG module revisions, features, and deviations, conformance can be more simply stated in terms of YANG packages, with a set of modifications (e.g. additional modules, deviations, or features).

- To allow YANG schema to be specified in a concise way rather than having each server explicitly list all modules, revisions, and features. YANG package definitions can be defined in documents that are available offline, and accessible via a URL, rather than requiring explicit lists of modules to be shared between client and server. Hence, a YANG package must contain sufficient information to allow a client or server to precisely construct the schema associated with the package.

- To define a mainly linear versioned history of sets of modules versions that are known to work together. I.e. to help mitigate the problem where a client must manage devices from multiple vendors, and vendor A implements version 1.0.0 of module foo and version 2.0.0 of module bar, and vendor B implements version 2.0.0 of module foo and version 1.0.0 of module bar. For a client, trying to interoperate with multiple vendors, and many YANG modules, finding a consistent lowest common denominator set of YANG module versions may be difficult, if not impossible.

Protocol mechanisms of how clients can negotiate which packages or package versions are to be used for NETCONF/RESTCONF communications are outside the scope of this document, and are defined in [I-D.ietf-netmod-yang-ver-selection].

Finally, the package definitions proposed by this document are intended to be relatively basic in their definition and the functionality that they support. As industry gains experience using

YANG packages, the standard YANG mechanisms of updating, or augmenting YANG modules could also be used to extend the functionality supported by YANG packages, if required.

5. YANG Package Definition

This document specifies an approach to defining YANG packages that is different to either of the approaches described in the background.

A YANG package is a versioned organizational structure defining a set of related YANG modules, packages, features, and deviations. A YANG package collectively defines a YANG schema.

Each YANG package has a name that SHOULD end with the suffix "-pkg". Package names are normally expected to be globally unique, but in some cases the package name may be locally scoped to a server or device, as described in Section 5.5.

YANG packages are versioned using the same approaches described in [I-D.ietf-netmod-yang-module-versioning] and [I-D.ietf-netmod-yang-semver]. This is described in further detail in Section 5.2.

Each YANG package version, defines:

- some metadata about the package, e.g., description, tags, scoping, referential completeness, location information.
- a set of YANG modules, at particular revisions, that are implemented by servers that implement the package. The modules may contain deviations.
- a set of import-only YANG modules, at particular revisions, that are used 'import-only' by the servers that implement the package.
- a set of included YANG packages, at particular revisions, that are also implemented by servers that implement the package.
- a set of YANG module features that must be supported by servers that implement the package.

The structure for YANG package definitions uses existing YANG language statements, YANG Data Structure Extensions [I-D.ietf-netmod-yang-data-ext], and YANG Instance Data File Format [I-D.ietf-netmod-yang-instance-file-format].

YANG package definitions are available offline in YANG instance data files. Client applications can be designed to support particular package versions that they expect to interoperate with.

YANG package definitions are available from the server via augmentations to YANG Library [RFC8525]. Rather than client applications downloading the entire contents of YANG library to confirm that the server schema is compatible with the client, they can check, or download, a much shorter YANG package definition, and validate that it conforms to the expected schema.

YANG package definitions can also be used to define the schema associated with YANG instance data files holding other, e.g., non packages related, instance data.

5.1. Package definition rules

Packages are defined using the following rules:

1. A YANG package MAY represent a complete YANG schema or only part of a YANG schema with some module import dependencies missing, as described in Section 5.4.

2. Packages definitions are hierarchical. A package can include other packages. Only a single version of a package can be included, and conflicting package includes (e.g. from descendant package includes) MUST be explicitly resolved by indicating which version takes precedence, and which versions are being replaced.

3. For each module implemented by a package, only a single revision of that module MUST be implemented. Multiple revisions of a module MAY be listed as import-only dependencies.

4. The revision of a module listed in the package 'module' list supersedes any 'implemented' revision of the module listed in an included package module list. The 'replaces-revision' leaf-list is used to indicate which 'implemented' or 'import-only' module revisions are replaces by this module revision. This allows a package to explicitly resolve conflicts between implemented module revisions in included packages.

5. The 'replaces-revision' leaf-list in the 'import-only-module' list can be used to exclude duplicate revisions of import-only modules from included packages. Otherwise, the import-only-modules for a package are the import-only-modules from all included packages combined with any modules listed in the packages import-only-module list.

6. YANG packages definitions MAY include modules containing deviation statements, but those deviation statements MUST only be used in an RFC 7950 compatible way to indicate where a server, or class of servers, deviates from a published standard. Deviations MUST NOT be included in a package definition that is part of a published standard. See section 5.8.1 for further guidance on the use of deviations in YANG packages.

5.2. Package versioning

Individual versions of a YANG package are versioned using the "revision-label" scheme defined in section 3.3 of [I-D.ietf-netmod-yang-module-versioning].

5.2.1. Updating a package with a new version

Package compatibility is fundamentally defined by how the YANG schema between two package versions has changed.

When a package definition is updated, the version associated with the package MUST be updated appropriately, taking into consideration the scope of the changes as defined by the rules below.

A package definition SHOULD define the previous version of the package in the 'previous-version' leaf unless it is the initial version of the package. If the 'previous-version' leaf is provided then the package definition MUST set the 'nbc-changes' leaf if the new version is non-backwards-compatible with respect to the package version defined in the 'previous-version' leaf.

5.2.1.1. Non-Backwards-compatible changes

The following changes classify as non-backwards-compatible changes to a package definition:

- Changing an 'included-package' list entry to select a package version that is non-backwards-compatible to the prior package version, or removing a previously included package.

- Changing a 'module' or 'import-only-module' list entry to select a module revision that is non-backwards-compatible to the prior module revision, or removing a previously implemented module.

- Removing a feature from the 'mandatory-feature' leaf-list.

- Adding, changing, or removing a deviation that is considered a non-backwards-compatible change to the affected data node in the schema associated with the prior package version.

5.2.1.2. Backwards-compatible changes

The following changes classify as backwards-compatible changes to a package definition:

- Changing an 'included-package' list entry to select a package version that is backwards-compatible to the prior package version, or including a new package that does not conflict with any existing included package or module.
- Changing a 'module' or 'import-only-module' list entry to select a module revision that is backwards-compatible to the prior module revision, or including a new module to the package definition.
- Adding a feature to the 'mandatory-feature' leaf-list.
- Adding, changing, or removing a deviation that is considered a backwards-compatible change to the affected data node in the schema associated with the prior package version.

5.2.1.3. Editorial changes

The following changes classify as editorial changes to a package definition:

- Changing a 'included-package' list entry to select a package version that is classified as an editorial change relative to the prior package version.
- Changing a 'module' or 'import-only-module' list entry to select a module revision that is classified as an editorial change relative to the prior module revision.
- Any change to any metadata associated with a package definition that causes it to have a different checksum value.

5.2.2. YANG Semantic Versioning for packages

YANG Semantic Versioning [I-D.ietf-netmod-yang-semver] MAY be used as an appropriate type of revision-label for the package version leaf.

If the format of the leaf matches the 'yangver:version' type specified in ietf-yang-semver.yang, then the package version leaf MUST be interpreted as a YANG semantic version number.

For YANG packages defined by the IETF, YANG semantic version numbers MUST be used as the version scheme for YANG packages.

The rules for incrementing the YANG package version number are equivalent to the semantic versioning rules used to version individual YANG modules, defined in section 3.2 of [I-D.ietf-netmod-yang-semver], but use the rules defined previously in Section 5.2.1 to determine whether a change is classified as non-backwards-compatible, backwards-compatible, or editorial. Where available, the semantic version number of the referenced elements in the package (included packages or modules) can be used to help determine the scope of changes being made.

5.2.3. Revision history

YANG packages do not contain a revision history. This is because packages may have many revisions and a long revision history would bloat the package definition. By recursively examining the 'previous-version' leaf of a package definition, a full revision history (including where non-backwards-compatible changes have occurred) can be dynamically constructed, if all package versions are available.

5.3. Package conformance

YANG packages allows for conformance to be checked at a package level rather than requiring a client to download all modules, revisions, and deviations from the server to ensure that the datastore schema used by the server is compatible with the client.

YANG package conformance is analogous to how YANG [RFC7950] requires that servers either implement a module faithfully, or otherwise use deviations to indicate areas of non-conformance.

For a top level package representing a datastore schema, servers MUST implement the package definition faithfully, including all mandatory features.

Package definitions MAY modify the schema for directly or hierarchically included packages through the use of different module revisions or module deviations. If the schema of any included package is modified in a non-backwards-compatible way then it MUST be indicated by setting the 'nbc-modified' leaf to true.

5.3.1. Use of YANG semantic versioning

Using the YANG semantic versioning scheme for package version numbers and module revision labels can help with conformance. In the general case, clients should be able to determine the nature of changes between two package versions by comparing the version number.

This usually means that a client does not have to be restricted to working only with servers that advertise exactly the same version of a package in YANG library. Instead, reasonable clients should be able to interoperate with any server that supports a package version that is backwards compatible to version that the client is designed for, assuming that the client is designed to ignore operational values for unknown data nodes.

For example, a client coded to support 'foo' package at version 1.0.0 should interoperate with a server implementing 'foo' package at version 1.3.5, because the YANG semantic versioning rules require that package version 1.3.5 is backwards compatible to version 1.0.0.

This also has a relevance on servers that are capable of supporting version selection because they need not support every version of a YANG package to ensure good client compatibility. Choosing suitable minor versions within each major version number should generally be sufficient, particular if they can avoid non-backwards-compatible patch level changes.

5.3.2. Package checksums

Each YANG package definition may have a checksum associated with it to allow a client to validate that the package definition of the server matches the expected package definition without downloading the full package definition from the server.

The checksum for a package is calculated using the SHA-256 hash (XXX, reference) of the full file contents of the YANG package instance data file. This means that the checksum includes all whitespace and formatting, encoding, and all meta-data fields associated with the package and the instance data file).

The checksum for a module is calculated using the SHA-256 hash of the YANG module file definition. This means that the checksum includes all whitespace, formatting, and comments within the YANG module.

Packages that are locally scoped to a server may not have an offline instance data file available, and hence MAY not have a checksum.

The package definition allows URLs and checksums to be specified for all included packages, modules and submodules within the package definition. Checksums SHOULD be included in package definitions to validate the full integrity of the package.

On a server, package checksums SHOULD also be provided for the top level packages associated with the datastore schema.

5.3.3. The relationship between packages and datastores

As defined by NMDA [RFC8342], each datastore has an associated datastore schema. Sections 5.1 and 5.3 of NMDA defines further constraints on the schema associated with datastores. These constraints can be summarized thus:

- The schema for all conventional datastores is the same.
- The schema for non conventional configuration datastores (e.g., dynamic datastores) may completely differ (i.e. no overlap at all) from the schema associated with the conventional configuration datastores, or may partially or fully overlap with the schema of the conventional configuration datastores. A dynamic datastore, for example, may support different modules than conventional datastores, or may support a subset or superset of modules, features, or data nodes supported in the conventional configuration datastores. Where a data node exists in multiple datastore schema it has the same type, properties and semantics.
- The schema for the operational datastore is intended to be a superset of all the configuration datastores (i.e. includes all the schema nodes from the conventional configuration datastores), but data nodes can be omitted if they cannot be accurately reported. The operational datastore schema can include additional modules containing only config false data nodes, but there is no harm in including those modules in the configuration datastore schema as well.

Given that YANG packages represent a YANG schema, it follows that each datastore schema can be represented using packages. In addition, the schema for most datastores on a server are often closely related. Given that there are many ways that a datastore schema could be represented using packages, the following guidance provides a consistent approach to help clients understand the relationship between the different datastore schema supported by a device (e.g., which parts of the schema are common and which parts have differences):

- Any datastores (e.g., conventional configuration datastores) that have exactly the same datastore schema MUST use the same package definitions. This is to avoid, for example, the creation of a 'running-cfg' package and a separate 'intended-cfg' package that have identical schema.
- Common package definitions SHOULD be used for those parts of the datastore schema that are common between datastores, when those datastores do not share exactly the same datastore schema. E.g.,

if a substantial part of the schema is common between the conventional, dynamic, and operational datastores then a single common package can be used to describe the common parts, along with other packages to describe the unique parts of each datastore schema.

- YANG modules that do not contain any configuration data nodes SHOULD be included in the package for configuration datastores if that helps unify the package definitions.

- The packages for the operational datastore schema MUST include all packages for all configuration datastores, along with any required modules defining deviations to mark unsupported data nodes. The deviations MAY be defined directly in the packages defining the operational datastore schema, or in separate non referentially complete packages.

- The schema for a datastore MAY be represented using a single package or as the union of a set of compatible packages, i.e., equivalently to a set of non-conflicting packages being included together in an overarching package definition.

5.4. Schema referential completeness

A YANG package may represent a schema that is 'referentially complete', or 'referentially incomplete', indicated in the package definition by the 'complete' flag.

If all import statements in all YANG modules included in the package (either directly, or through included packages) can be resolved to a module revision defined with the YANG package definition, then the package is classified as referentially complete. Conversely, if one or more import statements cannot be resolved to a module specified as part of the package definition, then the package is classified as referentially incomplete.

A package that represents the exact contents of a datastore schema MUST always be referentially complete.

Referentially incomplete packages can be used, along with locally scoped packages, to represent an update to a device's datastore schema as part of an optional software hot fix. E.g., the base software is made available as a complete globally scoped package. The hot fix is made available as an incomplete globally scoped package. A device's datastore schema can define a local package that implements the base software package updated with the hot fix package.

Referentially incomplete packages could also be used to group sets of logically related modules together, but without requiring a fixed dependency on all imported 'types' modules (e.g., iana-if-types.yang), instead leaving the choice of specific revisions of 'types' modules to be resolved when the package definition is used.

5.5. Package name scoping and uniqueness

YANG package names can be globally unique, or locally scoped to a particular server or device.

5.5.1. Globally scoped packages

The name given to a package MUST be globally unique, and it MUST include an appropriate organization prefix in the name, equivalent to YANG module naming conventions.

Ideally a YANG instance data file defining a particular package version would be publicly available at one or more URLs.

5.5.2. Server scoped packages

Package definitions may be scoped to a particular server by setting the 'is-local' leaf to true in the package definition.

Locally scoped packages MAY have a package name that is not globally unique.

Locally scoped packages MAY have a definition that is not available offline from the server in a YANG instance data file.

5.6. Submodules packages considerations

As defined in [RFC7950] and [I-D.ietf-netmod-yang-semver], YANG conformance and versioning is specified in terms of particular revisions of YANG modules rather than for individual submodules.

However, YANG package definitions also include the list of submodules included by a module, primarily to provide a location of where the submodule definition can be obtained from, allowing a YANG schema to be fully constructed from a YANG package instance data file definition.

5.7. Package tags

[I-D.ietf-netmod-module-tags] defines YANG module tags as a mechanism to annotate a module definition with additional metadata. Tags MAY also be associated to a package definition via the 'tags' leaf-list.

The tags use the same registry and definitions used by YANG module tags.

5.8. YANG Package Usage Guidance

It is RECOMMENDED that organizations that publish YANG modules also publish YANG package definition that group and version those modules into units of related functionality. This increases interoperability, by encouraging implementations to use the same collections of YANG modules versions. Using packages also makes it easier to understand relationship between modules, and enables functionality to be described on a more abstract level than individual modules.

5.8.1. Use of deviations in YANG packages

[RFC7950] section 5.6.3 defines deviations as the mechanism to allow servers to indicate where they do not conform to a published YANG module that is being implemented.

In cases where implementations contain deviations from published packages, then those implementations SHOULD define a package that includes both the published packages and all modules containing deviations. This implementation specific package accurately reflects the schema used by the device and allows clients to determine how the implementation differs from the published package schema in an offline consumable way, e.g., when published in an instance data file (see section 6).

Organizations may wish to reuse YANG modules and YANG packages published by other organizations for new functionality. Sometimes, they may desire to modify the published YANG modules. However, they MUST NOT use deviations in an attempt to achieve this because such deviations cause two problems:

- They prevent implementations from reporting their own deviations for the same nodes.

- They fracture the ecosystem by preventing implementations from conforming to the standards specified by both organizations. This hurts the interoperability in the YANG community, promotes development of disconnected functional silos, and hurts creativity in the market.

5.8.2. Use of features in YANG modules and YANG packages

The YANG language supports feature statements as the mechanism to make parts of a schema optional. Published standard YANG modules SHOULD make use of appropriate feature statements to provide flexibility in how YANG modules may be used by implementations and used by YANG modules published by other organizations.

YANG packages support 'mandatory features' which allow a package to specify features that MUST be implemented by any conformant implementation of the package as a mechanism to simplify and manage the schema represented by a YANG package.

5.9. YANG package core definition

The ietf-yang-package-types.yang module defines a grouping to specify the core elements of the YANG package structure that is used within YANG package instance data files (ietf-yang-package-instance.yang) and also on the server (ietf-yang-packages.yang).

The "ietf-yang-package-types" YANG module has the following structure:

```
module: ietf-yang-package-types

  grouping yang-pkg-identification-leafs
    +-- name       pkg-name
    +-- version    pkg-version

  grouping yang-pkg-instance
    +-- name                   pkg-name
    +-- version                pkg-version
    +-- timestamp?             yang:date-and-time
    +-- organization?          string
    +-- contact?               string
    +-- description?           string
    +-- reference?             string
    +-- complete?              boolean
    +-- local?                 boolean
    +-- previous-version?      pkg-version
    +-- nbc-changes?           boolean
    +-- tag*                   tags:tag
    +-- mandatory-feature*     scoped-feature
    +-- included-package* [name version]
    |  +-- name                 pkg-name
    |  +-- version              pkg-version
    |  +-- replaces-version*    pkg-version
```

```
      |  +-- nbc-modified?       boolean
      |  +-- location*           inet:uri
      |  +-- checksum?           pkg-types:sha-256-hash
      +-- module* [name]
      |  +-- name                 yang:yang-identifier
      |  +-- revision?            rev:revision-date-or-label
      |  +-- replaces-revision*   rev:revision-date-or-label
      |  +-- namespace?           inet:uri
      |  +-- location*            inet:uri
      |  +-- checksum?            pkg-types:sha-256-hash
      |  +-- submodule* [name]
      |     +-- name?       yang:yang-identifier
      |     +-- revision    yang:revision-identifier
      |     +-- location*   inet:uri
      |     +-- checksum?   pkg-types:sha-256-hash
      +-- import-only-module* [name revision]
         +-- name?                yang:yang-identifier
         +-- revision?            rev:revision-date-or-label
         +-- replaces-revision*   rev:revision-date-or-label
         +-- namespace?           inet:uri
         +-- location*            inet:uri
         +-- checksum?            pkg-types:sha-256-hash
         +-- submodule* [name]
            +-- name?       yang:yang-identifier
            +-- revision    yang:revision-identifier
            +-- location*   inet:uri
            +-- checksum?   pkg-types:sha-256-hash
```

6. Package Instance Data Files

YANG packages SHOULD be available offline from the server, defined as YANG instance data files [I-D.ietf-netmod-yang-instance-file-format] using the YANG schema below to define the package data.

The following rules apply to the format of the YANG package instance files:

1. The file SHOULD be encoded in JSON.

2. The name of the file SHOULD follow the format "<package-name>@<version>.json".

3. The package name MUST be specified in both the instance-data-set 'name' and package 'name' leafs.

4. The 'description' field of the instance-data-set SHOULD be "YANG package definition".

5. The 'timestamp', "organization', 'contact' fields are defined in both the instance-data-set metadata and the YANG package metadata. Package definitions SHOULD only define these fields as part of the package definition. If any of these fields are populated in the instance-data-set metadata then they MUST contain the same value as the corresponding leaves in the package definition.

6. The 'revision' list in the instance data file SHOULD NOT be used, since versioning is handled by the package definition.

7. The instance data file for each version of a YANG package SHOULD be made available at one of more locations accessible via URLs. If one of the listed locations defines a definitive reference implementation for the package definition then it MUST be listed as the first entry in the list.

The "ietf-yang-package" YANG module has the following structure:

```
module: ietf-yang-package

  structure package:
    // Uses the yang-package-instance grouping defined in
    // ietf-yang-package-types.yang
    +-- name                  pkg-name
    +-- version               pkg-version
    ... remainder of yang-package-instance grouping ...
```

7. Package Definitions on a Server

7.1. Package List

A top level 'packages' container holds the list of all versions of all packages known to the server. Each list entry uses the common package definition, but with the addition of package location and checksum information that cannot be contained within a offline package definition contained in an instance data file.

The '/packages/package' list MAY include multiple versions of a particular package. E.g. if the server is capable of allowing clients to select which package versions should be used by the server.

7.2. Tree diagram

The "ietf-yang-packages" YANG module has the following structure:

```
module: ietf-yang-packages
  +--ro packages
     +--ro package* [name version]
        // Uses the yang-package-instance grouping defined in
        // ietf-yang-package-types.yang, with location and checksum:
        +--ro name                  pkg-name
        +--ro version               pkg-version
        ... remainder of yang-package-instance grouping ...
        +--ro location*             inet:uri
        +--ro checksum?             pkg-types:sha-256-hash
```

8. YANG Library Package Bindings

The YANG packages module also augments YANG library to allow a server to optionally indicate that a datastore schema is defined by a package, or a union of compatible packages. Since packages can generally be made available offline in instance data files, it may be sufficient for a client to only check that a compatible version of the package is implemented by the server without fetching either the package definition, or downloading and comparing the full list of modules and enabled features.

If a server indicates that a datastore schema maps to a particular package, then it MUST exactly match the schema defined by that package, taking into account enabled features and any deviations.

If a server cannot faithfully implement a package then it can define a new package to accurately report what it does implement. The new package can include the original package as an included package, and the new package can define additional modules containing deviations to the modules in the original package, allowing the new package to accurately describe the server's behavior. There is no specific mechanism provided to indicate that a mandatory-feature in package definition is not supported on a server, but deviations MAY be used to disable functionality predicated by an if-feature statement.

The "ietf-yl-packages" YANG module has the following structure:

```
module: ietf-yl-packages
  augment /yanglib:yang-library/yanglib:schema:
    +--ro package* [name version]
       +--ro name        -> /pkgs:packages/package/name
       +--ro version     leafref
       +--ro checksum?   leafref
```

9. YANG packages as schema for YANG instance data document

YANG package definitions can be used as the schema definition for YANG instance data files. When using a package schema, the name and version of the package MUST be specified, a package checksum and/or URL to the package definition MAY also be provided.

The "ietf-yang-inst-data-pkg" YANG module has the following structure:

```
module: ietf-yang-inst-data-pkg

  augment-structure /yid:instance-data-set/yid:content-schema-spec:
    +--:(pkg-schema)
      +-- pkg-schema
        +-- name        pkg-name
        +-- version     pkg-version
        +-- location*   inet:uri
        +-- checksum?   pkg-types:sha-256-hash
```

10. YANG Modules

The YANG module definitions for the modules described in the previous sections.

```
<CODE BEGINS> file "ietf-yang-package-types@2020-01-21.yang"
module ietf-yang-package-types {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-package-types";
  prefix "pkg-types";

  import ietf-yang-revisions {
    prefix rev;
    reference "XXXX: Updated YANG Module Revision Handling";
```

```
  }

  import ietf-yang-types {
    prefix yang;
    rev:revision-or-derived 2019-07-21;
    reference "RFC 6991bis: Common YANG Data Types.";
  }

  import ietf-inet-types {
    prefix inet;
    rev:revision-or-derived 2013-07-15;
    reference "RFC 6991: Common YANG Data Types.";
  }

  import ietf-module-tags {
    prefix tags;
    // RFC Ed. Fix revision once revision date of
    // ietf-module-tags.yang is known.
    reference "RFC XXX: YANG Module Tags.";
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";

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

     Author:   Rob Wilton
               <mailto:rwilton@cisco.com>";

  description
    "This module provides type and grouping definitions for YANG
     packages.

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

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

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

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

    // RFC Ed.: update the date below with the date of RFC publication
    // and remove this note.
    // RFC Ed.: replace XXXX with actual RFC number and remove this
    // note.
    revision 2020-01-21 {
      rev:revision-label 0.2.0;
      description
        "Initial revision";
      reference
        "RFC XXXX: YANG Packages";
    }


    /*
     * Typedefs
     */

    typedef pkg-name {
      type yang:yang-identifier;
      description
        "Package names are typed as YANG identifiers.";
    }

    typedef pkg-version {
      type rev:revision-date-or-label;
      description
        "Package versions SHOULD be a revision-label (e.g. perhaps a
         YANG Semver version string).  Package versions MAY also be a
         revision-date";

    }

    typedef pkg-identifier {
      type rev:name-revision;
      description
        "Package identifiers combine a pkg-name and a pkg-version";
    }

    typedef scoped-feature {
      type string {
        pattern '[a-zA-Z_][a-zA-Z0-9\-_.]*:[a-zA-Z_][a-zA-Z0-9\-_.]*';
      }
```

```
      description
        "Represents a feature name scoped to a particular module,
         identified as the '<module-name>:<feature-name>', where both
         <module-name> and <feature-name> are YANG identifier strings,
         as defiend by Section 12 or RFC 6020.";
      reference
        "RFC XXXX, YANG Packages.";
    }

    typedef sha-256-hash {
      type string {
        length "64";
        pattern "[0-9a-fA-F]*";
      }
      description
        "A SHA-256 hash represented as a hexadecimal string.

         Used as the checksum for modules, submodules and packages in a
         YANG package definition.

         For modules and submodules the SHA-256 hash is calculated on
         the contents of the YANG file defining the module/submodule.

         For packages the SHA-256 hash is calculated on the file
         containing the YANG instance data document holding the package
         definition";
    }


    /*
     * Groupings
     */
    grouping yang-pkg-identification-leafs {
      description
        "Parameters for identifying a specific version of a YANG
         package";

      leaf name {
        type pkg-name;
        mandatory true;
        description
          "The YANG package name.";
      }

      leaf version {
        type pkg-version;
        mandatory true;
        description
```

```
            "Uniquely identies a particular version of a YANG package.

             Follows the definition for revision labels defined in
             draft-verdt-nemod-yang-module-versioning, section XXX";
        }
      }

      grouping yang-pkg-instance {
        description
          "Specifies the data node for a full YANG package instance
           represented either on a server or as a YANG instance data
           document.";
        uses yang-pkg-identification-leafs;

        leaf timestamp {
          type yang:date-and-time;

          description
            "An optional timestamp for when this package was created.
             This does not need to be unique across all versions of a
             package.";
        }

        leaf organization {
          type string;

          description "Organization responsible for this package";
        }

        leaf contact {
          type string;

          description
            "Contact information for the person or organization to whom
             queries concerning this package should be sent.";
        }

        leaf description {
          type string;

          description "Provides a description of the package";
        }

        leaf reference {
          type string;

          description "Allows for a reference for the package";
        }
```

```
      leaf complete {
        type boolean;
        default true;
        description
          "Indicates whether the schema defined by this package is
           referentially complete.  I.e. all module imports can be
           resolved to a module explicitly defined in this package or
           one of the included packages.";
      }

      leaf local {
        type boolean;
        default false;
        description
          "Defines that the package definition is local to the server,
           and the name of the package MAY not be unique, and the
           package definition MAY not be available in an offline file.

           Local packages can be used when the schema for the device
           can be changed at runtime through the addition or removal of
           software packages, or hot fixes.";
      }

      leaf previous-version {
        type pkg-version;
        description
          "The previous package version that this version has been
           derived from.  This leaf allows a full version history graph
           to be constructed if required.";
      }

      leaf nbc-changes {
        type boolean;
        default false;
        description
          "Indicates whether the defined package version contains
           non-backwards-compatible changes relative to the package
           version defined in the 'previous-version' leaf.";
      }

      leaf-list tag {
        type tags:tag;
        description
          "Tags associated with a YANG package.  Module tags defined in
           XXX, ietf-netmod-module-tags can be used here but with the
           modification that the tag applies to the entire package
           rather than a specific module.  See the IANA 'YANG Module
           Tag Prefix' registry for reserved prefixes and the IANA
```

```
          'YANG Module IETF Tag' registry for IETF standard tags.";
      }

      leaf-list mandatory-feature {
        type scoped-feature;
        description
          "Lists features from any modules included in the package that
           MUST be supported by any server implementing the package.

           Features already specified in a 'mandatory-feature' list of
           any included package MUST also be supported by server
           implementations and do not need to be repeated in this list.

           All other features defined in modules included in the
           package are OPTIONAL to implement.

           Features are identified using <module-name>:<feature-name>";
      }

      list included-package {
        key "name version";
        description
          "An entry in this list represents a package that is included
           as part of the package definition, or an indirectly included
           package that is changed in a non backwards compatible way.

           It can be used to resolve inclusion of conflicting package
           versions by explicitly specifying which package version is
           used.

           If included packages implement different revisions or
           versions of the same module, then an explicit entry in the
           module list MUST be provided to select the specific module
           version 'implemented' by this package definition.

           If the schema for any packages that are included, either
           directly or indirectly via another package include, are
           changed in any non-backwards-compatible way then they MUST
           be explicitly listed in the included-packages list with the
           'nbc-modified' leaf set to true.

           For import-only modules, the 'replaces-revision' leaf-list
           can be used to select the specific module versions used by
           this package.";
        reference
          "XXX";

        uses yang-pkg-identification-leafs;
```

```
        leaf-list replaces-version {
          type pkg-version;
          description
            "Gives the version of an included package version that
             is replaced by this included package revision.";
        }

        leaf nbc-modified {
          type boolean;
          default false;
          description
            "Set to true if any data nodes in this package are modified
             in a non backwards compatible way, either through the use
             of deviations, or because one of the modules has been
             replaced by an incompatible revision.  This could also
             occur if a module's revision was replaced by an earlier
             revision that had the effect of removing some data
             nodes.";
        }

        leaf-list location {
          type inet:uri;
          description
            "Contains a URL that represents where an instance data file
             for this YANG package can be found.

             This leaf will only be present if there is a URL available
             for retrieval of the schema for this entry.

             If multiple locations are provided, then the first
             location in the leaf-list MUST be the definitive location
             that uniquely identifies this package";
        }

        leaf checksum {
          type pkg-types:sha-256-hash;
          description
            "The SHA-256 hash calculated on the textual package
             definition, represented as a hexadecimal string.";
        }
      }

      list module {
        key "name";
        description
          "An entry in this list represents a module that must be
           implemented by a server implementing this package, as per
           RFC 7950 section 5.6.5, with a particular set of supported
```

```
           features and deviations.

           A entry in this list overrides any module revision
           'implemented' by an included package.  Any replaced module
           revision SHOULD also be listed in the 'replaces-revision'
           list.";
        reference
          "RFC 7950: The YANG 1.1 Data Modeling Language.";

        leaf name {
          type yang:yang-identifier;
          mandatory true;
          description
            "The YANG module name.";
        }

        leaf revision {
          type rev:revision-date-or-label;
          description
            "The YANG module revision date or revision-label.

             If no revision statement is present in the YANG module,
             this leaf is not instantiated.";
        }

        leaf-list replaces-revision {
          type rev:revision-date-or-label;
          description
            "Gives the revision of an module (implemented or
             import-only) defined in an included package that is
             replaced by this implemented module revision.";
        }

        leaf namespace {
          type inet:uri;
          description
            "The XML namespace identifier for this module.";
        }

        leaf-list location {
          type inet:uri;
          description
            "Contains a URL that represents the YANG schema resource
             for this module.

             This leaf will only be present if there is a URL available
             for retrieval of the schema for this entry.";
        }
```

```
        leaf checksum {
          type pkg-types:sha-256-hash;
          description
            "The SHA-256 hash calculated on the textual module
             definition, represented as a hexadecimal string.";
        }

        list submodule {
          key "name";
          description
            "Each entry represents one submodule within the
             parent module.";

          leaf name {
            type yang:yang-identifier;
            description
              "The YANG submodule name.";
          }

          leaf revision {
            type yang:revision-identifier;
            mandatory true;
            description
              "The YANG submodule revision date.  If the parent module
               include statement for this submodule includes a revision
               date then it MUST match this leaf's value.";
          }

          leaf-list location {
            type inet:uri;
            description
              "Contains a URL that represents the YANG schema resource
               for this submodule.

               This leaf will only be present if there is a URL
               available for retrieval of the schema for this entry.";
          }

          leaf checksum {
            type pkg-types:sha-256-hash;
            description
              "The SHA-256 hash calculated on the textual submodule
               definition, represented as a hexadecimal string.";
          }
        }
      }

      list import-only-module {
```

```
        key "name revision";
        description
          "An entry in this list indicates that the server imports
           reusable definitions from the specified revision of the
           module, but does not implement any protocol accessible
           objects from this revision.

           Multiple entries for the same module name MAY exist.  This
           can occur if multiple modules import the same module, but
           specify different revision-dates in the import statements.";

        leaf name {
          type yang:yang-identifier;
          description
            "The YANG module name.";
        }

        leaf revision {
          type rev:revision-date-or-label;
          description
            "The YANG module revision date or revision-label.

             If no revision statement is present in the YANG module,
             this leaf is not instantiated.";
        }

        leaf-list replaces-revision {
          type rev:revision-date-or-label;
          description
            "Gives the revision of an import-only-module defined in an
             included package that is replaced by this
             import-only-module revision.";
        }

        leaf namespace {
          type inet:uri;
          description
            "The XML namespace identifier for this module.";
        }

        leaf-list location {
          type inet:uri;
          description
            "Contains a URL that represents the YANG schema resource
             for this module.

             This leaf will only be present if there is a URL available
             for retrieval of the schema for this entry.";
```

```
        }

        leaf checksum {
          type pkg-types:sha-256-hash;
          description
            "The SHA-256 hash calculated on the textual submodule
             definition, represented as a hexadecimal string.";
        }

        list submodule {
          key "name";
          description
            "Each entry represents one submodule within the
             parent module.";

          leaf name {
            type yang:yang-identifier;
            description
              "The YANG submodule name.";
          }

          leaf revision {
            type yang:revision-identifier;
            mandatory true;
            description
              "The YANG submodule revision date.  If the parent module
               include statement for this submodule includes a revision
               date then it MUST match this leaf's value.";
          }

          leaf-list location {
            type inet:uri;
            description
              "Contains a URL that represents the YANG schema resource
               for this submodule.

               This leaf will only be present if there is a URL
               available for retrieval of the schema for this entry.";
          }

          leaf checksum {
            type pkg-types:sha-256-hash;
            description
              "The SHA-256 hash calculated on the textual submodule
               definition, represented as a hexadecimal string.";
          }
        }
      }
```

```
    }
  }
  <CODE ENDS>


  <CODE BEGINS> file "ietf-yang-package-instance@2020-01-21.yang"
  module ietf-yang-package-instance {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-yang-package-instance";
    prefix pkg-inst;

    import ietf-yang-revisions {
      prefix rev;
      reference "XXXX: Updated YANG Module Revision Handling";
    }

    import ietf-yang-package-types {
      prefix pkg-types;
      rev:revision-or-derived 0.2.0;
      reference "RFC XXX: YANG Schema Versioning.";
    }

    import ietf-yang-structure-ext {
      prefix sx;
      reference "RFC XXX: YANG Data Structure Extensions.";
    }

    organization
      "IETF NETMOD (Network Modeling) Working Group";

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

       Author:   Rob Wilton
                 <mailto:rwilton@cisco.com>";

    description
      "This module provides a definition of a YANG package, which is
       used as the schema for an YANG instance data document specifying
       a YANG package.

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

       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
```

```
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

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

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

    // RFC Ed.: update the date below with the date of RFC publication
    // and remove this note.
    // RFC Ed.: replace XXXX with actual RFC number and remove this
    // note.
    revision 2020-01-21 {
      rev:revision-label 0.2.0;
      description
        "Initial revision";
      reference
        "RFC XXXX: YANG Packages";
    }


    /*
     * Top-level structure
     */

    sx:structure package {
      description
        "Defines the YANG package structure for use in a YANG instance
         data document.";

      uses pkg-types:yang-pkg-instance;
    }
  }
  <CODE ENDS>



  <CODE BEGINS> file "ietf-yang-package@2020-01-21.yang"
  module ietf-yang-packages {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-yang-packages";
    prefix pkgs;
```

```
  import ietf-yang-revisions {
    prefix rev;
    reference "XXXX: Updated YANG Module Revision Handling";
  }

  import ietf-yang-package-types {
    prefix pkg-types;
    rev:revision-or-derived 0.2.0;
    reference "RFC XXX: YANG Packages.";
  }

  import ietf-inet-types {
    prefix inet;
    rev:revision-or-derived 2013-07-15;
    reference "RFC 6991: Common YANG Data Types.";
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";

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

     Author:   Rob Wilton
               <mailto:rwilton@cisco.com>";

  description
    "This module defines YANG packages on a server implementation.

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

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

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

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

```
     // RFC Ed.: update the date below with the date of RFC publication
     // and remove this note.
     // RFC Ed.: replace XXXX with actual RFC number and remove this
     // note.
     revision 2020-01-21 {
       rev:revision-label 0.2.0;
       description
         "Initial revision";
       reference
         "RFC XXXX: YANG Packages";
     }


     /*
      * Groupings
      */

     grouping yang-pkg-ref {
       description
         "Defines the leaves used to reference a single YANG package";

       leaf name {
         type leafref {
           path '/pkgs:packages/pkgs:package/pkgs:name';
         }
         description
           "The name of the references package.";
       }

       leaf version {
         type leafref {
           path '/pkgs:packages'
             + '/pkgs:package[pkgs:name = current()/../name]'
             + '/pkgs:version';
         }

         description
           "The version of the referenced package.";
       }

       leaf checksum {
         type leafref {
           path '/pkgs:packages'
             + '/pkgs:package[pkgs:name = current()/../name]'
             + '[pkgs:version = current()/../version]/pkgs:checksum';
         }

         description
```

```
            "The checksum of the referenced package.";
        }
      }

      grouping yang-ds-pkg-ref {
        description
          "Defines the list used to reference a set of YANG packages that
           collectively represent a datastore schema.";

        list package {
          key "name version";

          description
            "Identifies the YANG packages that collectively defines the
             schema for the associated datastore.

             The datastore schema is defined as the union of all
             referenced packages, that MUST represent a referentially
             complete schema.

             All of the referenced packages must be compatible with no
             conflicting module versions or dependencies.";

          uses yang-pkg-ref;
        }
      }


      /*
       * Top level data nodes.
       */

      container packages {
        config false;
        description "All YANG package definitions";

        list package {
          key "name version";

          description
            "YANG package instance";

          uses pkg-types:yang-pkg-instance;

          leaf-list location {
            type inet:uri;
            description
              "Contains a URL that represents where an instance data file
```

```
              for this YANG package can be found.

              This leaf will only be present if there is a URL available
              for retrieval of the schema for this entry.

              If multiple locations are provided, then the first
              location in the leaf-list MUST be the definitive location
              that uniquely identifies this package";
          }

          leaf checksum {
            type pkg-types:sha-256-hash;
            description
              "The checksum of the package this schema relates to,
               calculated on the 'YANG instance data file' package
               definition available in the 'location' leaf list.

               This leaf MAY be omitted if the referenced package is
               locally scoped without an associated checksum.";
          }
        }
      }
    }
    <CODE ENDS>



    <CODE BEGINS> file "ietf-yl-package@2020-01-21.yang"
    module ietf-yl-packages {
      yang-version 1.1;
      namespace "urn:ietf:params:xml:ns:yang:ietf-yl-packages";
      prefix yl-pkgs;

      import ietf-yang-revisions {
        prefix rev;
        reference "XXXX: Updated YANG Module Revision Handling";
      }

      import ietf-yang-packages {
        prefix pkgs;
        rev:revision-or-derived 0.2.0;
        reference "RFC XXX: YANG Packages.";
      }

      import ietf-yang-library {
        prefix yanglib;
        rev:revision-or-derived 2019-01-04;
        reference "RFC 8525: YANG Library";
```

```
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";

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

     Author:   Rob Wilton
               <mailto:rwilton@cisco.com>";

  description
    "This module provides defined augmentations to YANG library to
     allow a server to report YANG package information.

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

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

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

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


  // RFC Ed.: update the date below with the date of RFC publication
  // and remove this note.
  // RFC Ed.: replace XXXX with actual RFC number and remove this
  // note.
  revision 2020-01-21 {
    rev:revision-label 0.2.0;
    description
      "Initial revision";
    reference
      "RFC XXXX: YANG Packages";
  }
```

```
    /*
     * Augmentations
     */

    augment "/yanglib:yang-library/yanglib:schema" {
      description
        "Allow datastore schema to be related to a set of YANG
         packages";

      uses pkgs:yang-ds-pkg-ref;
    }
  }
  <CODE ENDS>



  <CODE BEGINS> file "ietf-yang-inst-data-pkg@2020-01-21.yang"
  module ietf-yang-inst-data-pkg {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-yang-inst-data-pkg";
    prefix yid-pkg;

    import ietf-yang-revisions {
      prefix rev;
      reference "XXXX: Updated YANG Module Revision Handling";
    }

    import ietf-yang-package-types {
      prefix pkg-types;
      rev:revision-or-derived 0.2.0;
      reference "RFC XXX: YANG Schema Versioning.";
    }

    import ietf-yang-structure-ext {
      prefix sx;
      reference "RFC XXX: YANG Data Structure Extensions.";
    }

    import ietf-yang-instance-data {
      prefix yid;
      reference "RFC XXX: YANG Instance Data File Format.";
    }

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

```
     organization
       "IETF NETMOD (Network Modeling) Working Group";

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

        Author:   Rob Wilton
                  <mailto:rwilton@cisco.com>";

     description
       "The module augments ietf-yang-instance-data to allow package
        definitions to be used to define schema in YANG instance data
        documents.

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

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

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

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

     // RFC Ed.: update the date below with the date of RFC publication
     // and remove this note.
     // RFC Ed.: replace XXXX with actual RFC number and remove this
     // note.
     revision 2020-01-21 {
       rev:revision-label 0.2.0;
       description
         "Initial revision";
       reference
         "RFC XXXX: YANG Packages";
     }

     /*
      * Augmentations
```

```
      */

     sx:augment-structure
       "/yid:instance-data-set/yid:content-schema-spec" {
       description
         "Add package reference to instance data set schema
          specification";
       case pkg-schema {
         container pkg-schema {
           uses pkg-types:yang-pkg-identification-leafs;

           leaf checksum {
             type pkg-types:sha-256-hash;
             description
               "The SHA-256 hash of the package, calculated on
                the textual package definition, represented as a
                hexadecimal string.";
           }

           leaf-list location {
             type inet:uri;
             description
               "Contains a URL that represents where an instance data
                file for this YANG package can be found.

                This leaf will only be present if there is a URL
                available for retrieval of the schema for this entry.

                If multiple locations are provided, then the first
                location in the leaf-list MUST be the definitive
                location that uniquely identifies this package";
           }
         }
       }
     }
   }
   <CODE ENDS>
```

11. Security Considerations

The YANG modules specified in this document defines a schema for data that is accessed by network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

Similarly to YANG library [I-D.ietf-netconf-rfc7895bis], some of the readable data nodes in these YANG modules may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes.

One additional key different to YANG library, is that the 'ietf-yang-package' YANG module defines a schema to allow YANG packages to be defined in YANG instance data files, that are outside the security controls of the network management protocols. Hence, it is important to also consider controlling access to these package instance data files to restrict access to sensitive information. SHA-256 checksums are used to ensure the integrity of YANG package definitions, imported modules, and sub-modules.

As per the YANG library security considerations, the module, revision and version information in YANG packages may help an attacker identify the server capabilities and server implementations with known bugs since the set of YANG modules supported by a server may reveal the kind of device and the manufacturer of the device. Server vulnerabilities may be specific to particular modules, module revisions, module features, or even module deviations. For example, if a particular operation on a particular data node is known to cause a server to crash or significantly degrade device performance, then the YANG packages information will help an attacker identify server implementations with such a defect, in order to launch a denial-of-service attack on the device.

12. IANA Considerations

It is expected that a central registry of standard YANG package definitions is required to support this solution.

It is unclear whether an IANA registry is also required to manage specific package versions. It is highly desirable to have a specific canonical location, under IETF control, where the definitive YANG package versions can be obtained from.

This document requests IANA to registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registrations are requested.

URI: urn:ietf:params:xml:ns:yang:ietf-yang-package-types.yang

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

   URI: urn:ietf:params:xml:ns:yang:ietf-yang-package-instance.yang
   Registrant Contact: The IESG.
   XML: N/A, the requested URI is an XML namespace.

   URI: urn:ietf:params:xml:ns:yang:ietf-yang-packages.yang
   Registrant Contact: The IESG.
   XML: N/A, the requested URI is an XML namespace.

   URI: urn:ietf:params:xml:ns:yang:ietf-yl-packages.yang
   Registrant Contact: The IESG.
   XML: N/A, the requested URI is an XML namespace.

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

This document requests that the following YANG modules are added in the "YANG Module Names" registry [RFC6020]:

```
   Name: ietf-yang-package-types.yang
   Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-package-
   types.yang
   Prefix: pkg-types
   Reference: RFC XXXX

   Name: ietf-yang-package-instance.yang
   Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-package-
   instance.yang
   Prefix: pkg-inst
   Reference: RFC XXXX

   Name: ietf-yang-packages.yang
   Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-packages.yang
   Prefix: pkgs
   Reference: RFC XXXX

   Name: ietf-yl-packages.yang
   Namespace: urn:ietf:params:xml:ns:yang:ietf-yl-packages.yang
   Prefix: yl-pkgs
   Reference: RFC XXXX

   Name: ietf-yang-inst-data-pkg.yang
   Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-inst-data-
   pkg.yang
   Prefix: yid-pkg
```

Reference: RFC XXXX

13. Open Questions/Issues

All issues, along with the draft text, are currently being tracked at https://github.com/rgwilton/YANG-Packages-Draft/issues/

14. Acknowledgements

Feedback helping shape this document has kindly been provided by Andy Bierman, James Cumming, Mahesh Jethanandani, Balazs Lengyel, Ladislav Lhotka,and Jan Lindblad.

15. References

15.1. Normative References

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

[I-D.ietf-netmod-module-tags]
Hopps, C., Berger, L., and D. Bogdanovic, "YANG Module Tags", draft-ietf-netmod-module-tags-10 (work in progress), February 2020.

[I-D.ietf-netmod-yang-data-ext]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Data Structure Extensions", draft-ietf-netmod-yang-data-ext-05 (work in progress), December 2019.

[I-D.ietf-netmod-yang-instance-file-format]
Lengyel, B. and B. Claise, "YANG Instance Data File Format", draft-ietf-netmod-yang-instance-file-format-12 (work in progress), April 2020.

[I-D.ietf-netmod-yang-module-versioning]
Wilton, R., Rahman, R., Lengyel, B., Clarke, J., Sterne, J., Claise, B., and K. D'Souza, "Updated YANG Module Revision Handling", draft-ietf-netmod-yang-module-versioning-01 (work in progress), July 2020.

[I-D.ietf-netmod-yang-semver]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "YANG Semantic Versioning", draft-ietf-netmod-yang-semver-01 (work in progress), July 2020.

[I-D.ietf-netmod-yang-solutions]
Wilton, R., "YANG Versioning Solution Overview", draft-ietf-netmod-yang-solutions-00 (work in progress), March 2020.

[I-D.ietf-netmod-yang-ver-selection]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and W. Bo, "YANG Schema Selection", draft-ietf-netmod-yang-ver-selection-00 (work in progress), March 2020.

[I-D.ietf-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", draft-ietf-netmod-yang-versioning-reqs-03 (work in progress), June 2020.

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

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

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

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

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

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

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

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

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

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

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

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

15.2. Informative References

[I-D.bierman-netmod-yang-package] Bierman, A., "The YANG Package Statement", draft-bierman-netmod-yang-package-00 (work in progress), July 2015.

[I-D.ietf-netmod-artwork-folding] Watsen, K., Auerswald, E., Farrel, A., and Q. WU, "Handling Long Lines in Inclusions in Internet-Drafts and RFCs", draft-ietf-netmod-artwork-folding-12 (work in progress), January 2020.

[openconfigsemver] "Semantic Versioning for OpenConfig Models", <http://www.openconfig.net/docs/semver/>.

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

Appendix A. Examples

This section provides various examples of YANG packages, and as such this text is non-normative. The purpose of the examples is to only illustrate the file format of YANG packages, and how package dependencies work. It does not imply that such packages will be

defined by IETF, or which modules would be included in those packages even if they were defined. For brevity, the examples exclude namespace declarations, and use a shortened URL of "tiny.cc/ietf-yang" as a replacement for "https://raw.githubusercontent.com/YangModels/yang/master/standard/ietf/RFC".

A.1. Example IETF Network Device YANG package

This section provides an instance data file example of an IETF Network Device YANG package formatted in JSON.

This example package is intended to represent the standard set of YANG modules, with import dependencies, to implement a basic network device without any dynamic routing or layer 2 services. E.g., it includes functionality such as system information, interface and basic IP configuration.

As for all YANG packages, all import dependencies are fully resolved. Because this example uses YANG modules that have been standardized before YANG semantic versioning, they modules are referenced by revision date rather than version number.

<CODE BEGINS> file "example-ietf-network-device-pkg.json"
========= NOTE: '\' line wrapping per BCP XX (RFC XXXX) ==========

```
 {
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-ietf-network-device-pkg",
    "pkg-schema": {
       package: "ietf-yang-package-defn-pkg@0.1.0.json"
    },
    "description": "YANG package definition",
    "content-data": {
      "ietf-yang-package-instance:yang-package": {
        "name": "example-ietf-network-device-pkg",
        "version": "1.1.2",
        "timestamp": "2018-12-13T17:00:00Z",
        "organization": "IETF NETMOD Working Group",
        "contact" : "WG Web:   <http://tools.ietf.org/wg/netmod/>, \
                     WG List:  <mailto:netmod@ietf.org>",
        "description": "Example IETF network device YANG package.\
           \
           This package defines a small sample set of \
           YANG modules that could represent the basic set of \
           modules that a standard network device might be expected \
           to support.",
```

```
          "reference": "XXX, draft-rwilton-netmod-yang-packages",
          "location": [ "file://example.org/yang/packages/\
                                  ietf-network-device@v1.1.2.json" ],
          "module": [
            {
              "name": "iana-crypt-hash",
              "revision": "2014-08-06",
              "location": [ "https://tiny.cc/ietf-yang/\
                                 iana-crypt-hash%402014-08-06.yang" ],
              "checksum": "fa9fde408ddec2c16bf2c6b9e4c2f80b\
                           813a2f9e48c127016f3fa96da346e02d"
            },
            {
              "name": "ietf-system",
              "revision": "2014-08-06",
              "location": [ "https://tiny.cc/ietf-yang/\
                                   ietf-system%402014-08-06.yang" ],
              "checksum": "8a692ee2521b4ffe87a88303a61a1038\
                           79ee26bff050c1b05a2027ae23205d3f"
            },
            {
              "name": "ietf-interfaces",
              "revision": "2018-02-20",
              "location": [ "https://tiny.cc/ietf-yang/\
                                 ietf-interfaces%402018-02-20.yang" ],
              "checksum": "f6faea9938f0341ed48fda93dba9a69a\
                           a32ee7142c463342efec3d38f4eb3621"
            },
            {
              "name": "ietf-netconf-acm",
              "revision": "2018-02-14",
              "location": [ "https://tiny.cc/ietf-yang/\
                                ietf-netconf-acm%402018-02-14.yang" ],
              "checksum": "e03f91317f9538a89296e99df3ff0c40\
                           03cdfea70bf517407643b3ec13c1ed25"
            },
            {
              "name": "ietf-key-chain",
              "revision": "2017-06-15",
              "location": [ "https://tiny.cc/ietf-yang/\
                                   ietf-key-chain@2017-06-15.yang" ],
              "checksum": "6250705f59fc9ad786e8d74172ce90d5\
                           8deec437982cbca7922af40b3ae8107c"
            },
            {
              "name": "ietf-ip",
              "revision": "2018-02-22",
              "location": [ "https://tiny.cc/ietf-yang/\
```

```
                                          ietf-ip%402018-02-22.yang" ],
              "checksum": "b624c84a66c128ae69ab107a5179ca8e\
                           20e693fb57dbe5cb56c3db2ebb18c894"
            }
            ],
            "import-only-module": [
              {
                "name": "ietf-yang-types",
                "revision": "2013-07-15",
                "location": [ "https://tiny.cc/ietf-yang/\
                                    ietf-yang-types%402013-07-15.yang" ],
                "checksum": "a04cdcc875764a76e89b7a0200c6b9d8\
                           00b10713978093acda7840c7c2907c3f"
              },
              {
                "name": "ietf-inet-types",
                "revision": "2013-07-15",
                "location": [ "https://tiny.cc/ietf-yang/\
                                    ietf-inet-types%402013-07-15.yang" ],
                "checksum": "12d98b0143a5ca5095b36420f9ebc1ff\
                           a61cfd2eaa850080244cadf01b86ddf9"
              }
            ]
          }
        }
      }
    }
    <CODE ENDS>
```

A.2. Example IETF Basic Routing YANG package

This section provides an instance data file example of a basic IETF Routing YANG package formatted in JSON.

This example package is intended to represent the standard set of YANG modules, with import dependencies, that builds upon the example-ietf-network-device YANG package to add support for basic dynamic routing and ACLs.

As for all YANG packages, all import dependencies are fully resolved. Because this example uses YANG modules that have been standardized before YANG semantic versioning, they modules are referenced by revision date rather than version number. Locations have been excluded where they are not currently known, e.g., for YANG modules defined in IETF drafts. In a normal YANG package, locations would be expected to be provided for all YANG modules.

```
   <CODE BEGINS> file "example-ietf-routing-pkg.json"
   ========== NOTE: '\' line wrapping per BCP XX (RFC XXXX) ===========

   {
     "ietf-yang-instance-data:instance-data-set": {
       "name": "example-ietf-routing-pkg",
       "module": [ "ietf-yang-package@2019-09-11.yang" ],
       "description": "YANG package definition",
       "content-data": {
         "ietf-yang-package-instance:yang-package": {
           "name": "example-ietf-routing",
           "version": "1.3.1",
           "timestamp": "2018-12-13T17:00:00Z",
           "description": "This package defines a small sample set of \
             IETF routing YANG modules that could represent the set of \
             IETF routing functionality that a basic IP network device \
             might be expected to support.",
           "reference": "XXX, draft-rwilton-netmod-yang-packages",
           "imported-packages": [
             {
               "name": "ietf-network-device",
               "version": "1.1.2",
               "location": [ "http://example.org/yang/packages/\
                                    ietf-network-device@v1.1.2.json" ],
               "checksum": ""
             }
           ],
           "module": [
             {
               "name": "ietf-routing",
               "revision": "2018-03-13",
               "location": [ "https://tiny.cc/ietf-yang/\
                                       ietf-routing@2018-03-13.yang" ],
               "checksum": ""
             },
             {
               "name": "ietf-ipv4-unicast-routing",
               "revision": "2018-03-13",
               "location": [ "https://tiny.cc/ietf-yang/\
                           ietf-ipv4-unicast-routing@2018-03-13.yang" ],
               "checksum": ""
             },
             {
               "name": "ietf-ipv6-unicast-routing",
               "revision": "2018-03-13",
               "location": [ "https://tiny.cc/ietf-yang/\
                           ietf-ipv6-unicast-routing@2018-03-13.yang" ],
               "checksum": ""
```

```
            },
            {
              "name": "ietf-isis",
              "revision": "2018-12-11",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            },
            {
              "name": "ietf-interfaces-common",
              "revision": "2018-07-02",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            },
            {
              "name": "ietf-if-l3-vlan",
              "revision": "2017-10-30",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            },
            {
              "name": "ietf-routing-policy",
              "revision": "2018-10-19",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            },
            {
              "name": "ietf-bgp",
              "revision": "2018-05-09",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            },
            {
              "name": "ietf-access-control-list",
              "revision": "2018-11-06",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            }
          ],
          "import-only-module": [
            {
              "name": "ietf-routing-types",
              "revision": "2017-12-04",
```

```
              "location": [ "https://tiny.cc/ietf-yang/\
                            ietf-routing-types@2017-12-04.yang" ],
              "checksum": ""
            },
            {
              "name": "iana-routing-types",
              "revision": "2017-12-04",
              "location": [ "https://tiny.cc/ietf-yang/\
                            iana-routing-types@2017-12-04.yang" ],
              "checksum": ""
            },
            {
              "name": "ietf-bgp-types",
              "revision": "2018-05-09",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            },
            {
              "name": "ietf-packet-fields",
              "revision": "2018-11-06",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            },
            {
              "name": "ietf-ethertypes",
              "revision": "2018-11-06",
              "location": [ "https://tiny.cc/ietf-yang/\
                            " ],
              "checksum": ""
            }
          ]
        }
      }
    }
  }
  <CODE ENDS>
```

A.3. Package import conflict resolution example

This section provides an example of how a package can resolve conflicting module versions from imported packages.

In this example, YANG package 'example-3-pkg' imports both 'example-import-1' and 'example-import-2' packages. However, the two imported packages implement different versions of 'example-module-A' so the

'example-3-pkg' package selects version '1.2.3' to resolve the conflict. Similarly, for import-only modules, the 'example-3-pkg' package does not require both versions of example-types-module-C to be imported, so it indicates that it only imports revision '2018-11-26' and not '2018-01-01'.

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-import-1-pkg",
    "description": "First imported example package",
    "content-data": {
      "ietf-yang-package-instance:yang-package": {
        "name": "example-import-1",
        "version": "1.0.0",
        "reference": "XXX, draft-rwilton-netmod-yang-packages",
        "revision-date": "2018-01-01",
        "module": [
          {
            "name": "example-module-A",
            "version": "1.0.0"
          },
          {
            "name": "example-module-B",
            "version": "1.0.0"
          }
        ],
        "import-only-module": [
          {
            "name": "example-types-module-C",
            "revision": "2018-01-01"
          },
          {
            "name": "example-types-module-D",
            "revision": "2018-01-01"
          }
        ]
      }
    }
  }
}

{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-import-2-pkg",
    "description": "Second imported example package",
    "content-data": {
      "ietf-yang-package:yang-package": {
```

```
          "name": "example-import-2",
          "version": "2.0.0",
          "reference": "XXX, draft-rwilton-netmod-yang-packages",
          "revision-date": "2018-11-26",
          "module": [
            {
              "name": "example-module-A",
              "version": "1.2.3"
            },
            {
              "name": "example-module-E",
              "version": "1.1.0"
            }
          ],
          "import-only-module": [
            {
              "name": "example-types-module-C",
              "revision": "2018-11-26"
            },
            {
              "name": "example-types-module-D",
              "revision": "2018-11-26"
            }
          ]
        }
      }
    }
  }

  {
    "ietf-yang-instance-data:instance-data-set": {
      "name": "example-3-pkg",
      "description": "Importing example package",
      "content-data": {
        "ietf-yang-package:yang-package": {
          "name": "example-3",
          "version": "1.0.0",
          "reference": "XXX, draft-rwilton-netmod-yang-packages",
          "revision-date": "2018-11-26",
          "included-package": [
            {
              "name": "example-import-1",
              "version": "1.0.0"
            },
            {
              "name": "example-import-2",
              "version": "2.0.0"
            }
```

```
          ],
          "module": [
            {
              "name": "example-module-A",
              "version": "1.2.3"
            }
          ],
          "import-only-module": [
            {
              "name": "example-types-module-C",
              "revision": "2018-11-26",
              "replaces-revision": [ "2018-01-01 "]
            }
          ]
        }
      }
    }
  }
```

Appendix B. Possible alternative solutions

This section briefly describes some alternative solutions. It can be removed if this document is adopted as a WG draft.

B.1. Using module tags

Module tags have been suggested as an alternative solution, and indeed that can address some of the same requirements as YANG packages but not all of them.

Module tags can be used to group or organize YANG modules. However, this raises the question of where this tag information is stored. Module tags either require that the YANG module files themselves are updated with the module tag information (creating another versioning problem), or for the module tag information to be hosted elsewhere, perhaps in a centralize YANG Catalog, or in instance data files similar to how YANG packages have been defined in this draft.

One of the principle aims of YANG packages is to be a versioned object that defines a precise set of YANG modules versions that work together. Module tags cannot meet this aim without an explosion of module tags definitions (i.e. a separate module tag must be defined for each package version).

Module tags cannot support the hierachical scheme to construct YANG schema that is proposed in this draft.

B.2. Using YANG library

Another question is whether it is necessary to define new YANG modules to define YANG packages, and whether YANG library could just be reused in an instance data file. The use of YANG packages offers several benefits over just using YANG library:

1. Packages allow schema to be built in a hierarchical fashion. [I-D.ietf-netconf-rfc7895bis] only allows one layer of hierarchy (using module sets), and there must be no conflicts between module revisions in different module-sets.

2. Packages can be made available off the box, with a well defined unique name, avoiding the need for clients to download, and construct/check the entire YANG schema for each device. Instead they can rely on the named packages with secure checksums. YANG library's use of a 'content-id' is unique only to the device that generated them.

3. Packages may be versioned using a semantic versioning scheme, YANG library does not provide a schema level semantic version number.

4. For a YANG library instance data file to contain the necessary information, it probably needs both YANG library and various augmentations (e.g. to include each module's semantic version number), unless a new version of YANG library is defined containing this information. The module definition for a YANG package is specified to contain all of the ncessary information to solve the problem without augmentations

5. YANG library is designed to publish information about the modules, datastores, and datastore schema used by a server. The information required to construct an off box schema is not precisely the same, and hence the definitions might deviate from each other over time.

Authors' Addresses

Robert Wilton (editor)
Cisco Systems, Inc.

Email: rwilton@cisco.com

Reshad Rahman
Cisco Systems, Inc.

Email: rrahman@cisco.com

Joe Clarke
Cisco Systems, Inc.

Email: jclarke@cisco.com

Jason Sterne
Nokia

Email: jason.sterne@nokia.com

Bo Wu (editor)
Huawei

Email: lana.wubo@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 5 May 2021

R. Wilton
Cisco Systems, Inc.

1 November 2020

YANG Schema Comparison
draft-ietf-netmod-yang-schema-comparison-01

Abstract

This document specifies an algorithm for comparing two revisions of a YANG schema to determine the scope of changes, and a list of changes, between the revisions. The output of the algorithm can be used to help select an appropriate revision-label or YANG semantic version number for a new revision. This document defines a YANG extension that provides YANG annotations to help the tool accurately determine the scope of changes between two revisions.

Table of Contents

1. Introduction

Warning, this is an early (-00) draft with the intention of scoping the outline of the solution, hopefully for the WG to back the direction of the solution. Refinement of the solution details is expected, if this approach is accepted by the WG.

This document defines a solution to Requirement 2.2 in [I-D.ietf-netmod-yang-versioning-reqs]. Complementary documents provide a complete solution to the YANG versioning requirements, with the overall relationship of the solution drafts described in [I-D.ietf-netmod-yang-solutions].

YANG module 'revision-labels' [I-D.ietf-netmod-yang-module-versioning] and the use of YANG semantic version numbers [I-D.ietf-netmod-yang-semver] can be used to help manage and report changes between revisions of individual YANG modules.

YANG packages [I-D.ietf-netmod-yang-packages] along with YANG semantic version numbers can be used to help manage and report changes between revisions of YANG schema.

[I-D.ietf-netmod-yang-module-versioning] and [I-D.ietf-netmod-yang-packages] define how to classify changes between two module or package revisions, respectively, as backwards compatible or non-backwards-compatible. [I-D.ietf-netmod-yang-semver] refines the definition, to allow backwards compatible changes to be classified as 'minor changes' or 'editorial changes'.

'Revision-label's and YANG semantic version numbers, whilst being generally simple and helpful in the mainline revision history case, are not sufficient in all scenarios. For example, when comparing two revisions/versions on independent revision branches, without a direct ancestor relationship between the two revisions/versions. In this cases, an algorithmic comparison approach is beneficial.

In addition, the module revision history's 'nbc-changes' extension statement, and YANG semantic version numbers, effectively declare the worst case scenario. If any non-backwards-compatible changes are restricted to only parts of the module/schema that are not used by an operator, then the operator is able to upgrade, and effectively treat the differences between the two revisions/versions as backwards compatible because they are not materially impacted by the non-backwards-compatible changes.

Hence, this document defines algorithms that can be applied to revisions of YANG modules or versions of YANG schema (e.g., as represented by YANG packages), to determine the changes, and scope of changes between the revisions/versions.

For many YANG statements, programmatic tooling can determine whether the changes between the statements constitutes a backwards-compatible or non-backwards-compatible change. However, for some statements, it is not feasible for current tooling to determine whether the changes are backwards-compatible or not. For example, in the general case, tooling cannot determine whether the change in a YANG description statement causes a change in the semantics of a YANG data node. If the change is to fix a typo or spelling mistake then the change can be classified as an editorial backwards-compatible change. Conversely, if the change modifies the behavioral specification of the data node then the change would need to be classified as either a non editorial backwards-compatible change or a non-backwards-compatible change. Hence, extension statements are defined to annotate a YANG module with additional information to clarify the scope of changes in cases that cannot be determined by algorithmic comparison.

Open issues are tracked at https://github.com/netmod-wg/yang-ver-dt/issues, tagged with 'schema-comparison'.

2. Terminology and Conventions

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

This document uses terminology introduced in the YANG versioning requirements document [I-D.ietf-netmod-yang-versioning-reqs].

This document makes of the following terminology introduced in the YANG Packages [I-D.ietf-netmod-yang-packages]:

* YANG schema

In addition, this document defines the terminology:

* Change scope: Whether a change between two revisions is classified as non-backwards-compatible, backwards-compatible, or editorial.

3. Generic YANG schema tree comparison algorithm

The generic schema comparison algorithm works on any YANG schema. This could be a schema associated with an individual YANG module, or a YANG schema represented by a set of modules, e.g., specified by a YANG package.

The algorithm performs a recursive tree wise comparison of two revisions of a YANG schema, with the following behavior:

The comparison algorithm primarily acts on the parts of the schema defined by unique identifiers.

Each identifier is qualified with the name of the module that defines the identifier.

Identifiers in different namespaces (as defined in 6.2.1 or RFC 7950) are compared separately. E.g., 'features' are compared separately from 'identities'.

Within an identifier namespace, the identifiers are compared between the two schema revisions by qualified identifier name. The 'renamed-from' extension allow for a meaningful comparison where the name of the identifier has changed between revisions. The 'renamed-from' identifier parameter is only used when an identifier in the new schema revision cannot be found in the old schema revision.

YANG extensions, features, identities, typedefs are checked by comparing the properties defined by their YANG sub-statements between the two revisions.

YANG groupings, top-level data definition statements, rpcs, and notifications are checked by comparing the top level properties defined by their direct child YANG sub-statements, and also by recursively checking the data definition statements.

The rules specified in section 3 of [I-D.ietf-netmod-yang-module-versioning] determine whether the changes are backwards-compatible or non-backwards-compatible.

The rules specified in section 3.2 of [I-D.ietf-netmod-yang-packages] determine whether backwards-compatible changes are 'minor' or 'editorial'.

For YANG description", "must", and "when" statements, the "backwards-compatible" and "editorial" extension statements can be used to mark instances when the statements have changed in a backwards-compatible or editorial way. Since by default the comparison algorithm assumes that any changes in these statements are non-backwards-compatible. XXX, more info required here, since the revisions in the module history probably need to be available for this to work in the general branched revisions case.

Submodules are not relevant for schema comparison purposes, i.e. the comparison is performed after submodule resolution has been completed.

3.1. YANG module revision scope extension annotations

4. YANG module comparison algorithm

The schema comparison algorithm defined in Section 3 can be used to compare the schema for individual modules, but with the following modifications:

Changes to the module's metadata information (i.e. module level description, contact, organization, reference) should be checked (as potential editorial changes).

The module's revision history should be ignored from the comparison.

Changes to augmentations and deviations should be sorted by path and compared.

5. YANG schema comparison algorithms

5.1. Standard YANG schema comparison algorithm

The standard method for comparing two YANG schema versions is to individually compare the module revisions for each module implemented by the schema using the algorithm defined in Section 4 and then aggregating the results together:

* If all implemented modules in the schema have only changed in an editorial way then the schema is changed in an editorial way

* If all implemented modules in the schema have only been changed in an editorial or backwards-compatible way then the schema is changed in a backwards-compatible way

* Otherwise if any implemented module in the schema has been changed in a non-backwards-compatible way then the schema is changed in a non-backwards-compatible way.

The standard schema comparison method is the RECOMMENDED scheme to calculate the version number change for new versions of YANG packages, because it allows the package version to be calculated based on changes to implemented modules revision history (or YANG semantic version number if used to identify module revisions).

5.2. Filtered YANG schema comparison algorithm

Another method to compare YANG schema, that is less likely to report inconsequential differences, is to construct full schema trees for the two schema versions, directly apply a version of the comparison algorithm defined in Section 3. This may be particular useful when the schema represents a complete datastore schema for a server because it allows various filtered to the comparison algorithm to provide a more specific answer about what changes may impact a particular client.

The full schema tree can easily be constructed from a YANG package definition, or alternative YANG schema definition.

Controlled by input parameters to the comparison algorithm, the following parts of the schema trees can optionally be filtered during the comparison:

- All "grouping" statements can be ignored (after all "use" statements have been processed when constructing the schema).
- All module and submodule metadata information (i.e. module level description, contact, organization, reference) can be ignored.
- The comparison can be restricted to the set of features that are of interest (different sets of features may apply to each schema versions).
- The comparison can be restricted to the subset of data nodes, RPCs, notifications and actions, that are of interest (e.g., the subset actually used by a particular client), providing a more meaningful result.
- The comparison could filter out backwards-compatible 'editorial' changes.

In addition to reporting the overall scope of changes at the schema level, the algorithm output can also optionally generate a list of specific changes between the two schema, along with the classification of those individual changes.

6. Comparison tooling

'pyang' has some support for comparison two module revisions, but this is currently limited to a linear module history.

TODO, it would be helpful if there is reference tooling for schema comparison.

7. Module Versioning Extension YANG Modules

YANG module with extension statements for annotating NBC changes, revision label, status description, and importing by version.

```
<CODE BEGINS> file "ietf-yang-rev-annotations@2019-11-11.yang"
module ietf-yang-rev-annotations {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-rev-annotations";
  prefix rev-ext;

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

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

  description
    "This YANG 1.1 module contains extensions to annotation to YANG
     module with additional metadata information on the nature of
     changes between two YANG module revisions.

     XXX, maybe these annotations could also be included in
     ietf-yang-revisions?

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

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

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

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

  // RFC Ed.: update the date below with the date of RFC publication
```

```
  // and remove this note.
  // RFC Ed.: replace XXXX (inc above) with actual RFC number and
  // remove this note.

  revision 2019-11-11 {
    description
      "Initial version.";
    reference
      "XXXX: YANG Schema Comparison";
  }

  extension backwards-compatible {
    argument revision-date-or-label;
    description
      "Identifies a revision (by revision-label, or revision date if
       a revision-label is not available) where a
       backwards-compatible change has occurred relative to the
       previous revision listed in the revision history.

       The format of the revision-label argument MUST conform to the
       pattern defined for the ietf-yang-revisions
       revision-date-or-label typedef.

       The following YANG statements MAY have zero or more
       'rev-ext:non-backwards-compatible' statements:
           description
           must
           when

       Each YANG statement MUST only a have a single
       non-backwards-compatible, backwards-compatible, or editorial
       extension statement for a particular revision-label, or
       corresponding revision-date.";

    reference
      "XXXX: YANG Schema Comparison;
       Section XXX, XXX";
  }

  extension editorial {
    argument revision-date-or-label;
    description
      "Identifies a revision (by revision-label, or revision date if
       a revision-label is not available) where an editorial change
       has occurred relative to the previous revision listed in the
       revision history.

       The format of the revision-label argument MUST conform to the
```

```
          pattern defined for the ietf-yang-revisions
          revision-date-or-label typedef.

          The following YANG statements MAY have zero or more
          'rev-ext:non-backwards-compatible' statements:
              description

          Each YANG statement MUST only a have a single
          non-backwards-compatible, backwards-compatible, or editorial
          extension statement for a particular revision-label or
          corresponding revision-date.";

        reference
          "XXXX: YANG Schema Comparison;
           Section XXX, XXX";
      }

      extension renamed-from {
        argument yang-identifier;
        description
          "Specifies a previous name for this identifier.

           This can be used when comparing schema to optimize handling
           for data nodes that have been renamed rather than naively
           treated them as data nodes that have been deleted and
           recreated.

           The argument 'yang-identifier' MUST take the form of a YANG
           identifier, as defined in section 6.2 of RFC 7950.

           Any YANG statement that takes a YANG identifier as its
           argument MAY have a single 'rev-ext:renamed-from'
           sub-statement.

           TODO, we should also facilitate identifiers being moved into
           other modules, e.g. by supporting a module-name qualified
           identifier.";

        reference
          "XXXX: YANG Schema Comparison;
           Section XXX, XXX";
      }
    }
    <CODE ENDS>
```

8. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The following individuals are (or have been) members of the design team and have worked on the YANG versioning project:

* Balazs Lengyel
* Benoit Claise
* Bo Wu
* Ebben Aries
* Jason Sterne
* Joe Clarke
* Juergen Schoenwaelder
* Mahesh Jethanandani
* Michael Wang
* Qin Wu
* Reshad Rahman
* Rob Wilton

The ideas for a tooling based comparison of YANG module revisions was first described in [I-D.clacla-netmod-yang-model-update]. This document extends upon those initial ideas.

9. Security Considerations

The document does not define any new protocol or data model. There are no security impacts.

10. IANA Considerations

10.1. YANG Module Registrations

The following YANG module is requested to be registered in the "IANA Module Names" registry:

The ietf-yang-rev-annotations module:

Name: ietf-yang-rev-annotations

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-rev-annotations

Prefix: rev-ext

Reference: [RFCXXXX]

11. References

11.1. Normative References

[I-D.ietf-netmod-yang-module-versioning]
Wilton, R., Rahman, R., Lengyel, B., Clarke, J., Sterne, J., Claise, B., and K. D'Souza, "Updated YANG Module Revision Handling", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-module-versioning-01, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netmod-yang-module-versioning-01>.

[I-D.ietf-netmod-yang-packages]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and W. Bo, "YANG Packages", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-packages-00, 17 March 2020, <https://tools.ietf.org/html/draft-ietf-netmod-yang-packages-00>.

[I-D.ietf-netmod-yang-semver]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "YANG Semantic Versioning", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-semver-01, 13 July 2020, <https://tools.ietf.org/html/draft-ietf-netmod-yang-semver-01>.

[I-D.ietf-netmod-yang-solutions]
Wilton, R., "YANG Versioning Solution Overview", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-solutions-00, 19 March 2020, <https://tools.ietf.org/html/draft-ietf-netmod-yang-solutions-00>.

[I-D.ietf-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-versioning-reqs-03, 29 June 2020, <https://tools.ietf.org/html/draft-ietf-netmod-yang-versioning-reqs-03>.

[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

[I-D.clacla-netmod-yang-model-update] Claise, B., Clarke, J., Lengyel, B., and K. D'Souza, "New YANG Module Update Procedure", Work in Progress, Internet-Draft, draft-clacla-netmod-yang-model-update-06, 2 July 2018, <https://tools.ietf.org/html/draft-clacla-netmod-yang-model-update-06>.

Author's Address

Robert Wilton
Cisco Systems, Inc.

Email: rwilton@cisco.com

Network Working Group
Internet-Draft
Updates: 8407 (if approved)
Intended status: Standards Track
Expires: 25 August 2021

B. Claise
Huawei
J. Clarke, Ed.
R. Rahman
R. Wilton, Ed.
Cisco Systems, Inc.
B. Lengyel
Ericsson
J. Sterne
Nokia
K. D'Souza
AT&T

21 February 2021

YANG Semantic Versioning
draft-ietf-netmod-yang-semver-02

Abstract

This document specifies a scheme and guidelines for applying a modified set of semantic versioning rules to revisions of YANG modules. Additionally, this document defines a revision-label for this modified semver scheme.

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 25 August 2021.

Copyright Notice

Table of Contents

1. Introduction

[I-D.ietf-netmod-yang-module-versioning] puts forth a number of concepts relating to modified rules for updating modules, a means to signal when a new revision of a module has non-backwards-compatible (NBC) changes compared to its previous revision, and a versioning scheme that uses the revision history as a lineage for determining from where a specific revision of a YANG module is derived. Additionally, section 3.3 of [I-D.ietf-netmod-yang-module-versioning] defines a revision label which can be used as an overlay or alias to

provide additional context or an additional way to refer to a specific revision.

This document defines a revision-label scheme that uses modified [semver] rules for YANG artifacts (i.e., YANG modules and YANG packages [I-D.ietf-netmod-yang-packages]) as well as the revision label definition for using this scheme. The goal of this is to add a human readable version label that provides compatibility information for the YANG artifact without one needing to compare or parse its body. The label and rules defined herein represent the RECOMMENDED revision label scheme for IETF YANG artifacts.

2. Terminology and Conventions

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

Additionally, this document uses the following terminology:

* YANG artifact: YANG modules, YANG packages [I-D.ietf-netmod-yang-packages] , and YANG schema elements are examples of YANG artifacts for the purposes of this document.

3. YANG Semantic Versioning

This section defines YANG Semantic Versioning, explains how it is used with YANG artifacts, and the rules associated with changing an artifact's semantic version number when its contents are updated.

3.1. YANG Semantic Versioning Pattern

YANG artifacts that employ semantic versioning as defined in this document MUST use a version string (e.g., in revision-label or as a package version) that corresponds to the following pattern: X.Y.Z_COMPAT. Where:

* X, Y and Z are mandatory non-negative integers that are each less than 2147483647 (i.e., the maximum signed 32-bit integer value) and MUST NOT contain leading zeroes

* The '.' is a literal period (ASCII character 0x2e)

* The '_' is an optional single literal underscore (ASCII character 0x5f) and MUST only present if the following COMPAT element is included

* COMPAT, if it is specified, MUST be either the literal string "compatible" or the literal string "non_compatible"

Additionally, [semver] defines two specific types of metadata that may be appended to a semantic version string. Pre-release metadata MAY be appended to a semver string after a trailing '-' character. Build metadata MAY be appended after a trailing '+' character. If both pre-release and build metadata are present, then build metadata MUST follow pre-release metadata. While build metadata MUST be ignored by YANG semver parsers, pre-release metadata MUST be used during module development and MUST be considered base on Section 5 . Both pre-release and build metadata are allowed in order to support all of the [semver] rules. Thus, a version lineage that follows strict [semver] rules is allowed for a YANG artifact.

To signal the use of this versioning scheme, modules MUST set the revision-label-scheme extension as defined in [I-D.ietf-netmod-yang-module-versioning] to the identity "yang-semver". That identity value is defined in the ietf-yang-semver module below.

Additionally, this ietf-yang-semver module defines a typedef that formally specifies the syntax of the YANG semver version string.

3.2. Semantic Versioning Scheme for YANG Artifacts

This document defines the YANG semantic versioning scheme that is used for YANG artifacts that employ the YANG semver label. The versioning scheme has the following properties:

* The YANG semantic versioning scheme is extended from version 2.0.0 of the semantic versioning scheme defined at semver.org [semver] to cover the additional requirements for the management of YANG artifact lifecyles that cannot be addressed using the semver.org 2.0.0 versioning scheme alone.

* Unlike the [semver] versioning scheme, the YANG semantic versioning scheme supports updates to older versions of YANG artifacts, to allow for bug fixes and enhancements to artifact versions that are not the latest. However, it does not provide for the unlimited branching and updating of older revisions which are documented by the general rules in [I-D.ietf-netmod-yang-module-versioning] .

* YANG artifacts that follow the [semver] versioning scheme are fully compatible with implementations that understand the YANG semantic versioning scheme defined in this document.

* If updates are always restricted to the latest revision of the artifact only, then the version numbers used by the YANG semantic versioning scheme are exactly the same as those defined by the [semver] versioning scheme.

Every YANG module versioned using the YANG semantic versioning scheme specifies the module's semantic version number as the argument to the 'rev:revision-label' statement.

Because the rules put forth in [I-D.ietf-netmod-yang-module-versioning] are designed to work well with existing versions of YANG and allow for artifact authors to migrate to this scheme, it is not expected that all revisions of a given YANG artifact will have a semantic version label. For example, the first revision of a module may have been produced before this scheme was available.

YANG packages that make use of this semantic versioning scheme will have their semantic version as the value of the "revision_label" property.

As stated above, the YANG semver version number is expressed as a string of the form: 'X.Y.Z_COMPAT'; where X, Y, and Z each represent non-negative integers smaller than 2147483647 without leading zeroes, and _COMPAT represents an optional suffix of either "_compatible" or "_non_compatible".

* 'X' is the MAJOR version. Changes in the MAJOR version number indicate changes that are non-backwards-compatible to versions with a lower MAJOR version number.

* 'Y' is the MINOR version. Changes in the MINOR version number indicate changes that are backwards-compatible to versions with the same MAJOR version number, but a lower MINOR version number and no PATCH "_compatible" or "_non_compatible" modifier.

* 'Z_COMPAT' is the PATCH version and modifier. Changes in the PATCH version number can indicate editorial, backwards-compatible, or non-backwards-compatible changes relative to versions with the same MAJOR and MINOR version numbers, but lower PATCH version number, depending on what form modifier "_COMPAT" takes:

  - If the modifier string is absent, the change represents an editorial change. An editorial change is defined to be a change in the YANG artifact's content that does not affect the semantic meaning or functionality provided by the artifact in any way. Some examples include correcting a spelling mistake in the description of a leaf within a YANG module, non-

significant whitespace changes (e.g. realigning description statements, or changing indendation), or changes to YANG comments. Note: restructuring how a module uses, or does not use, submodules is treated as an editorial level change on the condition that there is no change in the module's semantic behavior due to the restructuring.

- If, however, the modifier string is present, the meaning is described below:

- "_compatible" - the change represents a backwards-compatible change

- "_non_compatible" - the change represents a non-backwards-compatible change

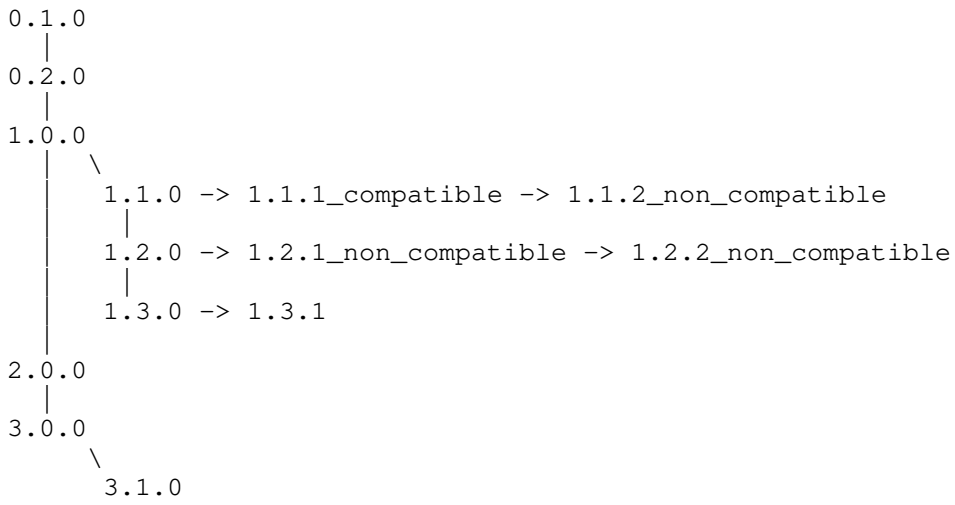
The YANG artifact name and YANG semantic version number uniquely identify a revision of said artifact. There MUST NOT be multiple instances of a YANG artifact definition with the same name and YANG semantic version number but different content (and in the case of modules, different revision dates).

There MUST NOT be multiple versions of a YANG artifact that have the same MAJOR, MINOR and PATCH version numbers, but different patch modifier strings. E.g., artifact version "1.2.3_non_compatible" MUST NOT be defined if artifact version "1.2.3" has already been defined.

3.2.1. Examples for YANG semantic version numbers

The following diagram and explanation illustrates how YANG semantic version numbers work.

Example YANG semantic version numbers for an example artifact:



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

0.1.0 - first beta module version

0.2.0 - second beta module version (with NBC changes)

1.0.0 - first release (may have NBC changes from 0.2.0)

1.1.0 - added new functionality, leaf "foo" (BC)

1.2.0 - added new functionality, leaf "baz" (BC)

1.3.0 - improve existing functionality, added leaf "foo-64" (BC)

1.3.1 - improve description wording for "foo-64" (Editorial)

1.1.1_compatible - backport "foo-64" leaf to 1.1.x to avoid implementing "baz" from 1.2.0 (BC)

2.0.0 - change existing model for performance reasons, e.g. re-key list (NBC)

1.1.2_non_compatible - NBC point bug fix, not required in 2.0.0 due to model changes (NBC)

3.0.0 - NBC bugfix, rename "baz" to "bar"; also add new BC leaf "wibble"; (NBC)

1.2.1_non_compatible - backport NBC fix, changing "baz" to "bar"

1.2.2_non_compatible - backport "wibble". This is a BC change but "non_compatible" modifier is sticky.

3.1.0 - introduce new leaf "wobble" (BC)

The partial ordering relationships based on the semantic versioning numbers can be defined as follows:

1.0.0 < 1.1.0 < 1.2.0 < 1.3.0 < 2.0.0 < 3.0.0 < 3.1.0

1.0.0 < 1.1.0 < 1.1.1_compatible < 1.1.2_non_compatible

1.0.0 < 1.1.0 < 1.2.0 < 1.2.1_non_compatible < 1.2.2_non_compatible

There is no ordering relationship between 1.1.1_non_compatible and either 1.2.0 or 1.2.1_non_compatible, except that they share the common ancestor of 1.1.0.

Looking at the version number alone, the module definition in 2.0.0 does not necessarily contain the contents of 1.3.0. However, the module revision history in 2.0.0 may well indicate that it was edited from module version 1.3.0.

3.3. YANG Semantic Version Update Rules

When a new revision of an artifact is produced, then the following rules define how the YANG semantic version number for the new artifact revision is calculated, based on the changes between the two artifact revisions, and the YANG semantic version number of the base artifact revision from which the changes are derived:

1. If an artifact is being updated in a non-backwards-compatible way, then the artifact version "X.Y.Z[_compatible|_non_compatible]" MUST be updated to "X+1.0.0" unless that artifact version has already been defined with different content, in which case the artifact version "X.Y.Z+1_non_compatible" MUST be used instead.

2. Under some circumstances (e.g., to avoid adding a "_compatible" modifier) an artifact author MAY also update the MAJOR version when the only changes are backwards-compatible. This is where tooling is important to highlight all changes. Because, while avoiding the "_compatible" and "_non_compatible" modifiers have a clear advantage, bumping a MAJOR version when changes are entirely backwards-compatible may confuse end users.

3. An artifact author MAY choose to skip version numbers. That is, an artifact's revision history can include 1.0.0, 1.1.0, and 1.3.0 where 1.2.0 is skipped. Note that doing so has an impact when importing modules by revision-or-derived. See Section 4 for more details on importing modules with revision-label version gaps.

4. If an artifact is being updated in a backwards-compatible way, then the next version number depends on the format of the current version number:

   i "X.Y.Z" - the artifact version MUST be updated to "X.Y+1.0", unless that artifact version has already been defined with different content, when the artifact version MUST be updated to "X.Y.Z+1_compatible"" instead.

   ii "X.Y.Z_compatible" - the artifact version MUST be updated to "X.Y.Z+1_compatible".

   iii "X.Y.Z_non_compatible" - the artifact version MUST be updated to "X.Y.Z+1_non_compatible".

5. If an artifact is being updated in an editorial way, then the next version number depends on the format of the current version number:

   i "X.Y.Z" - the artifact version MUST be updated to "X.Y.Z+1"

   ii "X.Y.Z_compatible" - the artifact version MUST be updated to "X.Y.Z+1_compatible".

   iii "X.Y.Z_non_compatible" - the artifact version MUST be updated to "X.Y.Z+1_non_compatible".

6. YANG artifact semantic version numbers beginning with 0, i.e "0.X.Y" are regarded as beta definitions and need not follow the rules above. Either the MINOR or PATCH version numbers may be updated, regardless of whether the changes are non-backwards-compatible, backwards-compatible, or editorial. See Section 5 for more details on using this notation during module development.

3.4. Examples of the YANG Semver Label

3.4.1. Example Module Using YANG Semver

Below is a sample YANG module that uses the YANG semver revision label based on the rules defined in this document.

```
module example-versioned-module {
  yang-version 1.1;
  namespace "urn:example:versioned:module";
  prefix "exvermod";
  rev:revision-label-scheme "yangver:yang-semver";

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

  description
    "to be completed";

  revision 2018-02-28 {
    description "Added leaf 'wobble'";
    rev:revision-label "3.1.0";
  }

  revision 2017-12-31 {
    description "Rename 'baz' to 'bar', added leaf 'wibble'";
    rev:revision-label "3.0.0";
    rev:nbc-changes;
  }

  revision 2017-10-30 {
    description "Change the module structure";
    rev:revision-label "2.0.0";
    rev:nbc-changes;
  }

  revision 2017-08-30 {
    description "Clarified description of 'foo-64' leaf";
    rev:revision-label "1.3.1";
  }

  revision 2017-07-30 {
    description "Added leaf foo-64";
    rev:revision-label "1.3.0";
  }

  revision 2017-04-20 {
    description "Add new functionality, leaf 'baz'";
    rev:revision-label "1.2.0";
  }

  revision 2017-04-03 {
    description "Add new functionality, leaf 'foo'";
    rev:revision-label "1.1.0";
  }
```

```
        revision 2017-04-03 {
          description "First release version.";
          rev:revision-label "1.0.0";
        }

        // Note: semver rules do not apply to 0.X.Y labels.

        revision 2017-01-30 {
          description "NBC changes to initial revision";
          semver:module-version "0.2.0";
        }

        revision 2017-01-26 {
          description "Initial module version";
          semver:module-version "0.1.0";
        }

        //YANG module definition starts here
```

3.4.2. Example of Package Using YANG Semver

Below is an example YANG package that uses the semver revision label based on the rules defined in this document.

```
   {
     "ietf-yang-instance-data:instance-data-set": {
       "name": "example-yang-pkg",
       "target-ptr": "TBD",
       "timestamp": "2018-09-06T17:00:00Z",
       "description": "Example IETF package definition",
       "content-data": {
         "ietf-yang-package:yang-package": {
           "name": "example-yang-pkg",
           "version": "1.3.1",
           ...
}
```

4. Import Module by Semantic Version

[I-D.ietf-netmod-yang-module-versioning] allows for imports to be done based on a module or a derived revision of a module. The rev:revision-or-derived statement can specify either a revision date or a revision label. When importing by semver, the YANG semver revision label value MAY be used as an argument to rev:revision-or-derived. In so, any module which has that semver label as its latest revision label or has that label in its revision history can be used to satisfy the import requirement. For example:

```
import example-module {
  rev:revision-or-derived "3.0.0";
}
```

Note: the import lookup does not stop when a non-backward-compatible change is encountered. That is, if module B imports a module A at or derived from version 2.0.0, resolving that import will pass through a revision of module A with version 2.1.0_non_compatible in order to determine if the present instance of module A derives from 2.0.0.

If an import by revision-or-derived cannot locate the specified revision-label in a given module's revision history, that import will fail. This is noted in the case of version gaps. That is, if a module's history includes 1.0.0, 1.1.0, and 1.3.0, an import from revision-or-derived at 1.2.0 will be unable to locate the specified revision entry and thus the import cannot be satisfied.

5. Guidelines for Using Semver During Module Development

This section and the IETF-specific sub-section below provides YANG semver-specific guidelines to consider when developing new YANG modules. As such this section updates [RFC8407] .

Development of a brand new YANG module outside of the IETF that uses YANG semver as its revision-label scheme SHOULD begin with a 0 for the MAJOR version component. This allows the module to disregard strict semver rules with respect to non-backwards-compatible changes during its initial development. However, module developers MAY choose to use the semver pre-release syntax instead with a 1 for the MAJOR version component. For example, an initial module revision-label might be either 0.0.1 or 1.0.0-alpha.1. If the authors choose to use the 0 MAJOR version component scheme, they MAY switch to the pre-release scheme with a MAJOR version component of 1 when the module is nearing initial release (e.g., a module's revision label may transition from 0.3.0 to 1.0.0-beta.1 to indicate it is more mature and ready for testing).

When using pre-release notation, the format MUST include at least one alphabetic component and MUST end with a '.' and then one or more digits. These alphanumeric components will be used when deciding pre-release precedence. The following are examples of valid pre-release versions

1.0.0-alpha.1

1.0.0-alpha.3

2.1.0-beta.42

3.0.0-202007.rc.1

When developing a new revision of an existing module using the YANG semver revision-label scheme, the intended target semver version MUST be used along with pre-release notation. For example, if a released module which has a current revision-label of 1.0.0 is being modified with the intent to make non-backwards-compatible changes, the first development MAJOR version component must be 2 with some pre-release notation such as -alpha.1, making the version 2.0.0-alpha.1. That said, every publicly available release of a module MUST have a unique YANG semver revision-label (where a publicly available release is one that could be implemented by a vendor or consumed by an end user). Therefore, it may be prudent to include the year or year and month development began (e.g., 2.0.0-201907-alpha.1). As a module undergoes development, it is possible that the original intent changes. For example, a 1.0.0 version of a module that was destined to become 2.0.0 after a development cycle may have had a scope change such that the final version has no non-backwards-compatible changes and becomes 1.1.0 instead. This change is acceptable to make during the development phase so long as pre-release notation is present in both versions (e.g., 2.0.0-alpha.3 becomes 1.1.0-alpha.4). However, on the next development cycle (after 1.1.0 is released), if again the new target release is 2.0.0, new pre-release components must be used such that every revision-label for a given module MUST be unique throughout its entire lifecycle (e.g., the first pre-release version might be 2.0.0-202005-alpha.1 if keeping the same year and month notation mentioned above).

5.1. Pre-release Version Precedence

[TODO: Describe precedence considering there could be changes during development and parallel development tracks.]

5.2. YANG Semver in IETF Modules

All publish IETF modules MUST use YANG semantic versions for their revision-labels. For IETF YANG modules that have already been published, revision labels MUST be retrospectively applied to all existing revisions when the next new revision is created, starting at version "1.0.0" for the initial published revision, and then incrementing according to the YANG Semver version rules specified in Section 3.3 .

Net new module development within the IETF SHOULD begin with the 0 MAJOR number scheme as described above. When revising an existing IETF module, the revision-label MUST use the target (i.e., intended) MAJOR and MINOR version components with a 0 PATCH version component. If the intended ratified release will be non-backward-compatible with the current ratified release, the MINOR version component MUST be 0.

All IETF modules in development MUST use the whole document name as a pre-release version string, including the current document revision. For example, if a module which is currently released at version 1.0.0 is being revised to include non-backwards-compatible changes in draft-user-netmod-foo, its development revision-labels MUST include 2.0.0-draft-user-netmod-foo followed by the document's revision (e.g., 2.0.0-draft-user-netmod-foo-02). This will ensure each pre-release version is unique across the lifecycle of the module. Even when using the 0 MAJOR version for initial module development (where MINOR and PATCH can change), appending the draft name as a pre-release component helps to ensure uniqueness when there are perhaps multiple, parallel efforts creating the same module.

If a module is being revised and the original module never had a revision-label (i.e., you wish to start using YANG semver in future module revisions), choose a semver value that makes the most sense based on the module's history. For example, if a module started out in the pre-NMDA ([RFC8342]) world, and then had NMDA support added without removing any legacy "state" branches -- and you are looking to add additional new features -- a sensible choice for the target YANG semver would be 1.2.0 (since 1.0.0 would have been the initial, pre-NMDA release, and 1.1.0 would have been the NMDA revision).

See Appendix A for a detailed example of IETF pre-release versions.

6. YANG Module

This YANG module contains the typedef for the YANG semantic version.

<CODE BEGINS> file "ietf-yang-semver@2020-06-30.yang"

```
module ietf-yang-semver {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-semver";
  prefix yangver;
  rev:revision-label-scheme "yang-semver";

  import ietf-yang-revisions {
    prefix rev;
  }

  organization
```

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

         Author:   Joe Clarke
                   <mailto:jclarke@cisco.com>";
      description
        "This module provides type and grouping definitions for YANG
         packages.

         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
         (http://trustee.ietf.org/license-info).

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

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

      revision 2020-06-30 {
        rev:revision-label "1.0.0-draft-ietf-netmod-yang-semver-01";
        description
          "Initial revision";
        reference
          "RFC XXXX: YANG Semantic Versioning.";
      }

      /*
       * Identities
       */

      identity yang-semver {
        base rev:revision-label-scheme-base-identity;
        description
          "The revision-label scheme corresponds to the YANG semver scheme
           which is defined by the pattern in the 'version' typedef below.
           The rules governing this revision-label scheme are defined in the
           reference for this identity.";
```

```
          reference
            "RFC XXXX: YANG Semantic Versioning.";
        }

        /*
         * Typedefs
         */

        typedef version {
          type string {
            pattern '\d+[.]\d+[.]\d+(_(non_)?compatible)?(-[\w\d.]+)?([+][\w\d\.]+
)?';
          }
          description
            "Represents a YANG semantic version number.  The rules governing the
             use of this revision label scheme are defined in the reference for
             this typedef.";
          reference
            "RFC XXXX: YANG Semantic Versioning.";
        }
      }
   <CODE ENDS>
```

7. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The design team consists of the following members whom have worked on the YANG versioning project:

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

* Rob Wilton

The initial revision of this document was refactored and built upon [I-D.clacla-netmod-yang-model-update] .

Discussons on the use of Semver for YANG versioning has been held with authors of the OpenConfig YANG models based on their own [openconfigsemver] . We would like thank both Anees Shaikh and Rob Shakir for their input into this problem space.

8. Security Considerations

The document does not define any new protocol or data model. There are no security impacts.

9. IANA Considerations

9.1. YANG Module Registrations

The following YANG module is requested to be registred in the "IANA Module Names" registry:

Name: ietf-yang-semver

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-semver

Prefix: yangver

Reference: [RFCXXXX]

9.2. Guidance for YANG Semver in IANA maintained YANG modules

Note for IANA (to be removed by the RFC editor): Please check that the registries and IANA YANG modules are referenced in the appropriate way.

IANA is responsible for maintaining and versioning some YANG modules, e.g., iana-if-types.yang [IfTypeYang] and iana-routing-types.yang [RoutingTypesYang] .

In addition to following the rules specified in the IANA Considerations section of [I-D.ietf-netmod-yang-module-versioning] , IANA maintained YANG modules MUST also include a YANG Semver revision label for all new revisions, as defined in Section 3 .

The YANG Semver version associated with the new revision MUST follow the rules defined in Section 3.3 .

Note: For IANA maintained YANG modules that have already been published, revision labels MUST be retrospectively applied to all existing revisions when the next new revision is created, starting at version "1.0.0" for the initial published revision, and then incrementing according to the YANG Semver version rules specified in Section 3.3 .

Most changes to IANA maintained YANG modules are expected to be backwards-compatible changes and classified as MINOR version changes. The PATCH version may be incremented instead when only editorial changes are made, and the MAJOR version would be incremented if non-backwards-compatible major changes are made.

Given that IANA maintained YANG modules are versioned with a linear history, it is anticipated that it should not be necessary to use the "_compatible" or "_non_compatible" modifiers to the "Z_COMPAT" version element.

10. References

10.1. Normative References

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

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

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[I-D.ietf-netmod-yang-module-versioning] Wilton, R., Rahman, R., Lengyel, B., Clarke, J., Sterne, J., Claise, B., and K. D'Souza, "Updated YANG Module Revision Handling", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-module-versioning-01, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netmod-yang-module-versioning-01>.

10.2. Informative References

[I-D.clacla-netmod-yang-model-update]
Claise, B., Clarke, J., Lengyel, B., and K. D'Souza, "New YANG Module Update Procedure", Work in Progress, Internet-Draft, draft-clacla-netmod-yang-model-update-06, 2 July 2018, <https://tools.ietf.org/html/draft-clacla-netmod-yang-model-update-06>.

[I-D.ietf-netmod-yang-packages]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and W. Bo, "YANG Packages", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-packages-01, 2 November 2020, <https://tools.ietf.org/html/draft-ietf-netmod-yang-packages-01>.

[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>.

[openconfigsemver]
"Semantic Versioning for Openconfig Models", <http://www.openconfig.net/docs/semver/>.

[semver] "Semantic Versioning 2.0.0", <https://www.semver.org>.

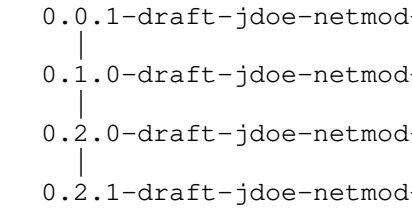
[IfTypeYang]
"iana-if-type YANG Module", <https://www.iana.org/assignments/iana-if-type/iana-if-type.xhtml>.

[RoutingTypesYang]
"iana-routing-types YANG Module", <https://www.iana.org/assignments/iana-routing-types/iana-routing-types.xhtml>.

Appendix A. Example IETF Module Development

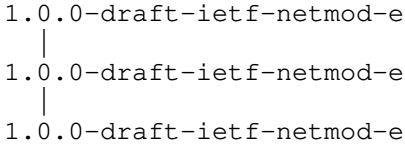
Assume a new YANG module is being developed in the netmod working group in the IETF. Initially, this module is being developed in an individual internet draft, draft-jdoe-netmod-example-module. The following represents the initial version tree (i.e., value of revision-label) of the module as it's being initially developed.

Version lineage for initial module development:



At this point, development stabilizes, and the workgroup adopts the draft. Thus now the draft becomes draft-ietf-netmod-example-module. The initial pre-release lineage continues as follows.

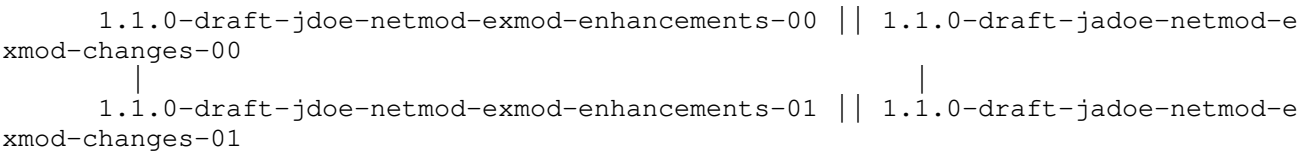
Continued version lineage after adoption:



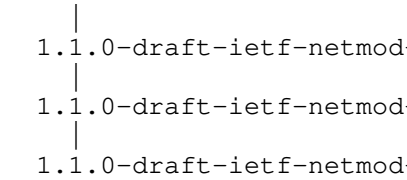
At this point, the draft is ratified and becomes RFC12345 and the YANG module version number becomes 1.0.0.

A time later, the module needs to be revised to add additional capabilities. Development will be done in a backwards-compatible way. Two new individual drafts are proposed to go about adding the capabilities in different ways: draft-jdoe-netmod-exmod-enhancements and draft-jadoe-netmod-exmod-changes. These are initially developed in parallel with the following versions.

Parallel development for next module revision:



At this point, the WG decides to merge some aspects of both and adopt the work in jadoe's draft as draft-ietf-netmod-exmod-changes. A single version lineage continues.



The draft is ratified, and the new module version becomes 1.1.0.

Authors' Addresses

Benoit Claise
Huawei

Email: benoit.claise@huawei.com

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

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

Reshad Rahman
Cisco Systems, Inc.

Email: rrahman@cisco.com

Robert Wilton (editor)
Cisco Systems, Inc.

Email: rwilton@cisco.com

Balazs Lengyel
Ericsson
1117 Budapest
Magyar Tudosok Korutja
Hungary

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

Jason Sterne
Nokia

Email: jason.sterne@nokia.com

Kevin D'Souza
AT&T
200 S. Laurel Ave
Middletown, NJ
United States of America

Email: kd6913@att.com