Internet Engineering Task Force (IETF)
Request for Comments: 8340
BCP: 215
Category: Best Current Practice
ISSN: 2070-1721

M. Bjorklund
Tail-f Systems
L. Berger, Ed.
LabN Consulting, L.L.C.

March 2018

YANG Tree Diagrams

Abstract

This document captures the current syntax used in YANG module tree diagrams. The purpose of this document is to provide a single location for this definition. This syntax may be updated from time to time based on the evolution of the YANG language.

Status of This Memo

This memo documents an Internet Best Current Practice.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on BCPs is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8340.

Table of Contents

1. Introduction 2
2. Tree Diagram Syntax 3
2.1. Submodules 5
2.2. Groupings 5
2.3. yang-data 5
2.4. Collapsed Node Representation 6
2.5. Comments 6
2.6. Node Representation 6
3. Usage Guidelines for RFCs 7
3.1. Wrapping Long Lines 8
3.2. Groupings 8
3.3. Long Diagrams 8
4. YANG Schema Mount Tree Diagrams 9
4.1. Representation of Mounted Schema Trees 10
5. IANA Considerations 12
6. Security Considerations 12
7. Informative References 12
Authors' Addresses 13

1. Introduction

YANG tree diagrams were first published in RFC 6536. Such diagrams are used to provide a simplified graphical representation of a data model and can be automatically generated via tools such as "pyang" [PYANG]. This document describes the syntax used in YANG tree diagrams. It is expected that this document will be updated or replaced as changes to the YANG language [RFC7950] necessitate.

Today's common practice is to include the definition of the syntax used to represent a YANG module in every document that provides a tree diagram. This practice has several disadvantages; therefore, the purpose of this document is to provide a single location for this definition. It is not the intent of this document to restrict future changes, but rather to ensure that such changes are easily identified and suitably agreed upon.

An example tree diagram can be found in Section 3 of [RFC8343]; the following is a portion of it:

```
  +--rw interfaces
     +--rw interface* [name]
        +--rw name                          string
        +--rw description?                  string
        +--rw type                          identityref
        +--rw enabled?                      boolean
        +--rw link-up-down-trap-enable?     enumeration {if-mib}?
```

2. Tree Diagram Syntax

This section describes the meaning of the symbols used in YANG tree diagrams.

A full tree diagram of a module represents all elements. It includes the name of the module and sections for top-level module statements (typically containers), augmentations, rpcs, and notifications all identified under a module statement. Module trees may be included in a document as a whole, by one or more sections, or even by subsets of nodes.

A module is identified by "module:" followed by the module-name. This is followed by one or more sections, in order:

1. The top-level data nodes defined in the module, offset by two spaces.

2. Augmentations, offset by two spaces and identified by the keyword "augment" followed by the augment target node and a colon (":") character.

3. RPCs, offset by two spaces and identified by "rpcs:".

4. Notifications, offset by two spaces and identified by "notifications:".

5. Groupings, offset by two spaces and identified by the keyword "grouping" followed by the name of the grouping and a colon (":") character.

6. yang-data, offset by two spaces and identified by the keyword "yang-data" followed by the name of the yang-data structure and a colon (":") character.

The relative organization of each section is provided using a text-based format that is typical of a file system directory tree display command. Each node in the tree is prefaced with "+--". Schema nodes that are children of another node are offset from the parent by three spaces. Sibling schema nodes are listed with the same space offset and, when separated by lines, are linked via a vertical bar ("|") character.

The full format, including spacing conventions, is:

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

    augment <target-node>:
      +--<node>
         +--<node>
         +--<node>
            +--<node>
    augment <target-node>:
      +--<node>

    rpcs:
      +--<rpc-node>
      +--<rpc-node>
         +--<node>
         |  +--<node>
         +--<node>

    notifications:
      +--<notification-node>
      +--<notification-node>
         +--<node>
         |  +--<node>
         +--<node>

    grouping <grouping-name>:
      +--<node>
         +--<node>
         |  +--<node>
         +--<node>
    grouping <grouping-name>:
      +--<node>

    yang-data <yang-data-name>:
      +--<node>
         +--<node>
         |  +--<node>
         +--<node>
    yang-data <yang-data-name>:
      +--<node>
```

2.1. Submodules

Submodules are represented in the same fashion as modules but are identified by "submodule:" followed by the (sub)module-name. For example:

```
  submodule: <module-name>
    +--<node>
    |  +--<node>
    |     +--<node>
```

2.2. Groupings

Nodes within a used grouping are normally expanded as if the nodes were defined at the location of the "uses" statement. However, it is also possible to not expand the "uses" statement but to instead print the name of the grouping.

For example, the following diagram shows the "tls-transport" grouping from [RFC7407] unexpanded:

```
    +--rw tls
       +---u tls-transport
```

If the grouping is expanded, it could be printed as:

```
    +--rw tls
       +--rw port?                 inet:port-number
       +--rw client-fingerprint?   x509c2n:tls-fingerprint
       +--rw server-fingerprint?   x509c2n:tls-fingerprint
       +--rw server-identity?      snmp:admin-string
```

Groupings may optionally be present in the "groupings" section.

2.3. yang-data

If the module defines a "yang-data" structure [RFC8040], these structures may optionally be present in the "yang-data" section.

2.4. Collapsed Node Representation

At times when the composition of the nodes within a module schema is not important in the context of the presented tree, sibling nodes and their children can be collapsed using the notation "..." in place of the text lines used to represent the summarized nodes. For example:

```
    +--<node>
    |  ...
    +--<node>
       +--<node>
          +--<node>
```

2.5. Comments

Single line comments, starting with "//" (possibly indented) and ending at the end of the line, may be used in the tree notation.

2.6. Node Representation

Each node in a YANG module is printed as:

```
  <status>--<flags> <name><opts> <type> <if-features>

    <status> is one of:
      +  for current
      x  for deprecated
      o  for obsolete

    <flags> is one of:
      rw  for configuration data nodes and choice nodes
      ro  for non-configuration data nodes and choice nodes,
          output parameters to rpcs and actions, and
          notification parameters
      -w  for input parameters to rpcs and actions
      -u  for uses of a grouping
      -x  for rpcs and actions
      -n  for notifications
      mp  for nodes containing a "mount-point" extension statement

      Case nodes do not have any <flags>.
```

```
      <name> is the name of the node
        (<name>) means that the node is a choice node
       :(<name>) means that the node is a case node

        If the node is augmented into the tree from another module,
        its name is printed as <prefix>:<name>, where <prefix> is the
        prefix defined in the module where the node is defined.

        If the node is a case node, there is no space before the
        <name>.

      <opts> is one of:
        ?  for an optional leaf, choice, anydata, or anyxml
        !  for a presence container
        *  for a leaf-list or list
        [<keys>] for a list's keys
        /  for a top-level data node in a mounted module
        @  for a top-level data node of a module identified in a
           mount point parent reference

      <type> is the name of the type for leafs and leaf-lists

        If the type is a leafref, the type is printed as either
        (1) "-> TARGET", where TARGET is the leafref path,
        with prefixes removed if possible or (2) "leafref".

      <if-features> is the list of features this node depends on,
        printed within curly brackets and a question mark "{...}?"
```

Arbitrary whitespace is allowed between any of the whitespace-separated fields (e.g., <opts> and <type>). Additional whitespace may, for example, be used to "column align" fields (e.g., within a list or container) to improve readability.

3. Usage Guidelines for RFCs

This section provides general guidelines related to the use of tree diagrams in RFCs.

3.1. Wrapping Long Lines

Internet-Drafts and RFCs limit the number of characters that may appear in a line of text to 72 characters. When the tree representation of a node results in a line being longer than this limit, the line should be broken between <opts> and <type> or between <type> and <if-feature>. The new line should be indented so that it starts below <name> with a whitespace offset of at least two characters. For example:

```
  notifications:
    +---n yang-library-change
       +--ro module-set-id
               -> /modules-state/module-set-id
```

Long paths (e.g., leafref paths or augment targets) can be split and printed on more than one line. For example:

```
  augment /nat:nat/nat:instances/nat:instance/nat:mapping-table
            /nat:mapping-entry:
```

The previously mentioned "pyang" command can be helpful in producing such output; for example, the notification diagram above was produced using:

```
  pyang -f tree --tree-line-length 50 ietf-yang-library.yang
```

When a tree diagram is included as a figure in an Internet-Draft or RFC, "--tree-line-length 69" works well.

3.2. Groupings

If the YANG module is comprised of groupings only, then the tree diagram should contain the groupings. The "pyang" compiler can be used to produce a tree diagram with groupings using the "-f tree --tree-print-groupings" command-line parameters.

3.3. Long Diagrams

Tree diagrams can be split into sections to correspond to document structure. As tree diagrams are intended to provide a simplified view of a module, diagrams longer than a page should generally be avoided. If the complete tree diagram for a module becomes too long, the diagram can be split into several smaller diagrams. For example, it might be possible to have one diagram with the data node and another with all notifications. If the data nodes tree is too long, it is also possible to split the diagram into smaller diagrams for

different subtrees. When long diagrams are included in a document, authors should consider whether to include the long diagram in the main body of the document or in an appendix.

An example of such a split can be found in [RFC7407], where Section 2.4 of that document shows the diagram for "engine configuration":

```
    +--rw snmp
       +--rw engine
          // more parameters from the "engine" subtree here
```

Further, Section 2.5 of [RFC7407] shows the diagram for "target configuration":

```
    +--rw snmp
       +--rw target* [name]
          // more parameters from the "target" subtree here
```

The previously mentioned "pyang" command can be helpful in producing such output; for example, the above example was produced using:

```
  pyang -f tree --tree-path /snmp/target ietf-snmp.yang
```

4. YANG Schema Mount Tree Diagrams

"YANG schema mount" is defined in [SCHEMA-MOUNT] and warrants some specific discussion. Schema mount is a generic mechanism that allows for the mounting of one or more YANG modules at a specified location of another (parent) schema. The specific location is referred to as a "mount point", and any container or list node in a schema may serve as a mount point. Mount points are identified via the inclusion of the "mount-point" extension statement as a substatement under a container or list node. Mount point nodes are thus directly identified in a module schema definition and can be identified in a tree diagram as indicated above using the "mp" flag.

In the following example taken from [YANG-NIs], "vrf-root" is a container that includes the "mount-point" extension statement as part of its definition:

```
  module: ietf-network-instance
    +--rw network-instances
       +--rw network-instance* [name]
          +--rw name           string
          +--rw enabled?       boolean
          +--rw description?   string
          +--rw (ni-type)?
          +--rw (root-type)
             +--:(vrf-root)
             |  +--mp vrf-root
```

4.1. Representation of Mounted Schema Trees

The actual modules made available under a mount point are controlled by a server and are provided to clients. This information is typically provided via the schema mount module ("ietf-yang-schema-mount") defined in [SCHEMA-MOUNT]. The schema mount module supports the exposure of both mounted schema and "parent-references". Parent references are used for XML Path Language (XPath) evaluation within mounted modules and do not represent client-accessible paths; the referenced information is available to clients via the parent schema. Schema mount also defines an "inline" type of mount point, where mounted modules are exposed via the YANG library module.

Although the modules made available under a mount point are not specified in YANG modules that include mount points, the document defining the module will describe the intended use of the module and may identify both modules that will be mounted and parent modules that can be referenced by mounted modules. An example of such a description can be found in [YANG-NIs]. A specific implementation of a module containing mount points will also support a specific list of mounted and referenced modules. In describing both intended use and actual implementations, it is helpful to show how mounted modules would be instantiated and referenced under a mount point using tree diagrams.

In such diagrams, the mount point should be treated much like a container that uses a grouping. The flags should also be set based on the "config" leaf mentioned above, and the mount-related options indicated above should be shown for the top-level nodes in a mounted or referenced module. The following example, taken from [YANG-NIs],

represents the prior example with the YANG modules "ietf-routing" [YANG-Routing] and "ietf-ospf" [OSPF-YANG] mounted, nodes from the YANG module "ietf-interfaces" [RFC8343] accessible via a parent-reference, and "config" indicating "true":

```
  module: ietf-network-instance
    +--rw network-instances
       +--rw network-instance* [name]
          +--rw name           string
          +--rw enabled?       boolean
          +--rw description?   string
          +--rw (ni-type)?
          +--rw (root-type)
             +--:(vrf-root)
                +--mp vrf-root
                   +--ro rt:routing-state/
                   |  +--ro router-id?
                   |  +--ro control-plane-protocols
                   |     +--ro control-plane-protocol* [type name]
                   |        +--ro ospf:ospf
                   |           +--ro instance* [af]
                   |           ...
                   +--rw rt:routing/
                   |  +--rw router-id?
                   |  +--rw control-plane-protocols
                   |     +--rw control-plane-protocol* [type name]
                   |     +--rw ospf:ospf
                   |        +--rw instance* [af]
                   |           ...
                   +--ro if:interfaces@
                   |  ...
                   +--ro if:interfaces-state@
                   |  ...
```

It is worth highlighting that the "ietf-ospf" module augments the "ietf-routing" module, and although it is listed in the schema mount module (or inline YANG library), there is no special mount-related notation in the tree diagram.

A mount point definition alone is not sufficient to identify whether the mounted modules are used for configuration data or for non-configuration data. This is determined by the "ietf-yang-schema-mount" module's "config" leaf associated with the specific mount point and is indicated on the top-level mounted nodes.

For example, in the above tree, when the "config" leaf for the "ietf-routing" module indicates "false", the nodes in the "rt:routing" subtree would have different flags:

```
+--ro rt:routing/
|  +--ro router-id?
|  +--ro control-plane-protocols
   ...
```

5. IANA Considerations

This document has no IANA actions.

6. Security Considerations

There is no security impact related to the tree diagrams defined in this document.

7. Informative References

[OSPF-YANG] Yeung, D., Qu, Y., Zhang, J., Chen, I., and A. Lindem, "Yang Data Model for OSPF Protocol", Work in Progress, draft-ietf-ospf-yang-10, March 2018.

[PYANG] "pyang", February 2018, <https://github.com/mbj4668/pyang>.

[RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <https://www.rfc-editor.org/info/rfc7407>.

[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>.

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

[SCHEMA-MOUNT] Bjorklund, M. and L. Lhotka, "YANG Schema Mount", Work in Progress, draft-ietf-netmod-schema-mount-08, October 2017.

[YANG-NIs] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Model for Network Instances", Work in Progress, draft-ietf-rtgwg-ni-model-11, March 2018.

[YANG-Routing] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", Work in Progress, draft-ietf-netmod-rfc8022bis-11, January 2018.

Authors' Addresses

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Lou Berger (editor)
LabN Consulting, L.L.C.

Email: lberger@labn.net