How to Generate YANG Annotation Modules

tail-f a Cisco company

Table of Contents

How to Generate YANG Annotation Modules

1. Introduction

YANG data models have really caught on, but when it comes to doing custom processing or manipulation of YANG files, there are not many tools out there other than pyang or yanger. While you can write custom plug-ins for pyang and yanger, it takes time and specialized knowledge of the tool.

This application note presents an innovative yet simple approach to processing YANG files using the extraction of Tail-f YANG extension statements as an example. The method presented leverages the fact that the YANG standard includes an XML mapping of YANG which is called YIN. It is possible to losslessly convert from YANG to YIN and back to YANG. Once a YANG file is converted to YIN, any of the many available XML processing tools can be used to manipulate the YIN model. When the desired processing is done, the YIN model can be converted back to a YANG model.

2. Tail-f ConfD YANG Extensions

When working with ConfD, developers often make use of Tail-f YANG extension statements in their YANG data models to extend ConfD and integrate applications. For example, when integrating backend instrumentation using "tailf:callpoint" statements, validation code integration using "tailf:validate" statements, or northbound CLI management interface auto-rendering customization using "tailf:cli-*" statements.

All these extensions are handled as normal YANG extensions. (YANG is designed to be extended). The Tail-f proprietary extensions are defined in the file $CONFD_DIR/src/confd/yang/tailf-common.yang.

Here is an example YANG data model with the Tail-f YANG extensions for a callpoint and a validation point:

```
module test {
  namespace "http://tail-f.com/test";
  prefix "t";
  import tailf-common {
    prefix tailf;
  }
  grouping mygrp {
    leaf a {
      type int32;
      tailf:callpoint mycp;
    }
    leaf b {
```

```
      when "../a";
      tailf:validate myvp {
        tailf:dependency "../a";
      }
      type string;
    }
  }
  container top {
    uses mygrp;
  }
}
```

Example 1 YANG data model with Tail-f YANG extensions

It is common and convenient for application developers to, at first, add Tail-f YANG extension statements directly in-line in the user's own proprietary or standard YANG data model module. However, when it comes time to provide the network element's YANG data models to clients, it is usually not necessary, or desirable, to keep the Tail-f YANG extension statements in line with the YANG models. The Tail-f YANG extensions relate to the internals of the system and are not needed by NETCONF or RESTCONF clients that make use of the YANG schema. Also, oftentimes, their presence can cause problems for the NETCONF client that does not support these Tail-f proprietary extensions, unless the client is for example Cisco NSO (Network Service Orchestrator) that can make use of, for example, tailf:dependency statements and CLI extensions. Even if the client understands the Tail-f YANG extensions, many extensions may have unintended consequences if kept in-line with the YANG model exposed to the clients.

3. Tail-f ConfD YANG Annotation Files

ConfD provides an alternative to using the Tail-f YANG extensions in-line in YANG data models through annotation modules using the tailf:annotate-module/statement or tailf:annotate Tail-f YANG extension statements. This feature allows the developer to separate the Tail-f YANG extension statements from the main YANG data model and place them into a separate annotation module (file) which the confdc compiler will merge during data model compilation when generating the database schema and northbound interfaces. Statements in annotation modules and the annotation modules themselves will not be exposed to NETCONF and RESTCONF clients that download the YANG schema or alternatively integrated YANG data models from bundles provided offline.

Here is the example YANG data model from the previous section, but using an annotation module instead:

```
module test {
  namespace "http://tail-f.com/test";
  prefix "t";
  import tailf-common {
    prefix tailf;
  }
  grouping mygrp {
    leaf a {
      type int32;
    }
    leaf b {
      when "../a";
```

```
      type string;
    }
  }
  container top {
    uses mygrp;
  }
}

module test-ann {
  namespace "http://tail-f.com/test-ann";
  prefix t-ann;

  import tailf-common {
    prefix tailf;
  }

  tailf:annotate-module "test" {
    tailf:annotate-statement "grouping[name='mygrp']" {
      tailf:annotate-statement "leaf[name='a']" {
        tailf:callpoint "mycp";
      }
    }
    tailf:annotate-statement "grouping[name='mygrp']" {
      tailf:annotate-statement "leaf[name='b']" {
        tailf:validate "myvp" {
          tailf:dependency "../a";
        }
      }
    }
  }
}
```

Example 2 Annotation module adding Tail-f YANG extensions to a YANG model

3.1 tailf:annotate-module/statement vs. tailf:annotate

ConfD provides two ways of annotating YANG data models:

- tailf:annotate-module/statement – Annotate YANG models as-is.
- tailf:annotate – Annotate the resulting database schema after compiling the YANG data models into the final schema.

The tailf:annotate-module/statement variant can be seen as more intuitive to use as the YANG data models are annotated, where it can be tricky to figure out the resulting schema after groupings, augments, choices, includes from submodules, etc. has been applied.

A key feature with the tailf:annotate-module/statement variant is that it is also possible to write tools to automate annotation file creation when using tailf:annotate-module/statement to annotate the YANG data models. This is not 100% possible when the tailf:annotate variant is used as some YANG data model information not used by the final schema is discarded when generating the database schema using confdc, NSO ncsc, or the yanger tool.

4. Automating the YANG Annotation Module Creation

For projects that start adding annotation statements in-line in YANG data models, the result is that the original YANG data models are populated with Tail-f extension annotations. Often with many of those YANG extension statements, performing the task of manually extracting them to annotation modules can be tedious and time-consuming. As an alternative, the annotation file can be created automatically. The key idea leveraged is that YANG can be converted to YIN, an XML representation of YANG, and back again. See the YANG 1.1 RFC for more on the YIN XML format. Once converted to XML, we can use existing XML tools for processing YANG statements.

```
<?xml version="1.0" encoding="UTF-8"?>
<module name="test"
        xmlns="urn:ietf:params:xml:ns:yang:yin:1"
        xmlns:t="http://tail-f.com/test"
        xmlns:tailf="http://tail-f.com/yang/common">
  <namespace uri="http://tail-f.com/test"/>
  <prefix value="t"/>
  <import module="tailf-common">
    <prefix value="tailf"/>
  </import>
  <grouping name="mygrp">
    <leaf name="a">
      <type name="int32"/>
      <tailf:callpoint id="mycp"/>
    </leaf>
    <leaf name="b">
      <when condition="../a"/>
      <tailf:validate id="myvp">
        <tailf:dependency path="../a"/>
      </tailf:validate>
      <type name="string"/>
    </leaf>
  </grouping>
  <container name="top">
    <uses name="mygrp"/>
  </container>
</module>
```

Example 3 YIN XML representation of the test.yang module

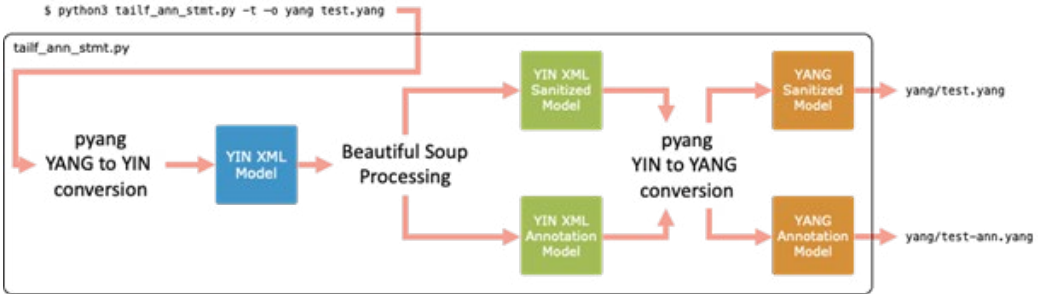
4.1 The ann-stmt Demo

A simple Python script can automate the work of creating an annotation file. A demo of such a script can be found in the ConfD-Developer GitHub repository under ConfD-Demos in the ann-stmt project. The ann-stmt project uses the pyang tool to convert YANG data models to YIN, and, once in XML format, the powerful Python Beautiful Soup + lxml libraries for processing YIN XML to extract the annotations from the original YANG model.

About the ann-stmt demo:

- The demo converts YANG data models to YIN XML using the pyang Python tool.
- The XML format is supported by many powerful tools that can aid in the modification of the YANG modules that are now in YIN format.

- This demo uses the powerful Python Beautiful Soup Python (bs4) library that in turn uses the lxml library for processing XML.
- After the tailf extension XML nodes have been extracted to tailf:annotate-module/ statement variants, pyang is used to convert the new annotation module and the original YIN module back to YANG modules.
- Optionally "must", "when", "min-elements", "max-elements", "mandatory", "unique", and "pattern" statements can be moved to the annotation module. This can be useful when debugging, for example, XPath evaluation, performance, etc., issues.
- "tailf:callpoint" and "tailf:validate" extensions can be removed without removing other tailf extensions. This can also be useful for debugging purposes.
- Just want to remove some statements and not create an annotation file? There is a flag for that too.
- See flags used with the tailf_ann_stmt.py Python code for all options.



The README file in the ann-stmt project in the ConfD-Developer GitHub repository provides the details on the setup and usage to automate the extraction of tailf statements and selected YANG statements for sanitizing YANG modules and creating a YANG annotation file or just sanitizing YANG modules for debugging purposes.

Note that you can also use the combination of Python BeautifulSoup BS4 and lxml for all sorts of XML processing. Not only for YANG data models when in YIN XML format but also XML representing for example ConfD or NSO configuration or operational state data can be modified with very little Python code and programming effort.

5. Summary

This application note has shown a simple yet powerful method for the custom processing or manipulation of YANG data models. This method of conversion from YANG to YIN, using XML processing tools, and then converting back to YANG can be used to accomplish much more than just the example presented in this application note.

6. For More Information

For more information about ConfD, visit https://www.tail-f.com

For more information about the tools used in this application note:

https://github.com/ConfD-Developer/ConfD-Demos/tree/master/ann-stmt
https://github.com/mbj4668/pyang
https://github.com/mbj4668/yanger

For more information about the standards mentioned in this application note:

The YANG 1.1 Data Modeling Language YIN section
NETCONF <get-schema> operation
RESTCONF schema resource

ConfD User Guide:

Section 5.7 Using the Tail-f Extensions with YANG

ConfD man pages:

tailf_yang_extensions(5): tailf:annotate-module statement
confdc(1): the -a, --annotate flag

tail-f a Cisco company

www.tail-f.com
info@tail-f.com

Corporate Headquarters

Sveavagen 25
111 34 Stockholm
Sweden
+46 8 21 37 40