1 SMIYang

1.1 Presentation

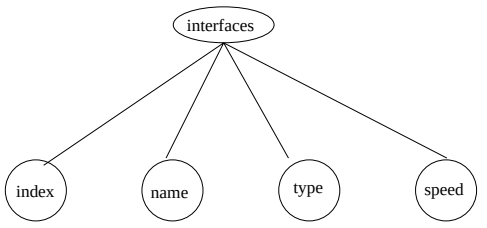
It is common in the network management world that a protocol and a data model are separated even if jointly designed, as it was already the case in the SNMP[4] protocol and its SMI[3] data modeling,COPS[6] and SPPI[8], or SMIng and various protocols [9] (GDMO and CMIS or WBEM and CIM outside the IETF scope).

NETCONF [7] is the IETF standard that emerged from the netconf working group to configure network devices. The netmod[1] working group defines YANG [1] as a candidate language to specify data models of values carried by NETCONF. This report describes a YANG parser called *jYang* that provides a syntaxic and semantic validation of YANG specifications (called modules or submodules).

A YANG specification contains formal definitions of data types that will model real data maintained by NETCONF agents. Formal definitions follow the YANG syntax. YANG provides constructs that give semantics to XML data that are exanged by NETCONF agents and managers. As an XML document is a collection of imbricated markups, YANG defines statements that can be mapped on pattern of markups. Moreover YANG allows reusability of specifications with generic statements or augmentation/extension statements.

Following is an example of a part of a YANG specification[2] that describes a table of network interfaces, a conceptual view of two entries and the XML document of this configuration:

```
list interfaces {
   key index;
   leaf index {
     type int8;
   }
   leaf name {
     type string;
   }
   leaf type {
     type string;
   }
   leaf speed {
     type int64;
   }
}
```




```
<list>
  <index>
     1
  </index>
  <name>
     loopback
  </name>
  <type>
     software-loopback
  </type>
  <speed>
     100000000
  </speed>
  <index>
     2
  </index>
  <name>
     ethernet
  </name>
  <type>
     ethernet-csmacd
  </type>
  <speed>
     100000000
  </speed>
</list>
```

[1] http://www.ietf.org/html.charters/netmod-charter.html
[2] All example in this report are inspired from the draft[1]

YANG specifications are organized in modules and submodules that contain data type definitions and operation descriptions.

YIN is an alternative XML-based syntax for YANG specifications. YIN specifications can be generated from YANG ones and are equivalent. The goal of YIN specifications is to enable seemless interactions with XML based tools (as XSLT). *jYang* parser allows the generation of YIN specifications from YANG.

2 *jYang*

jYang is a java parser for YANG specifications and an application programming interface offering a programmatic access in java to YANG specifications.

2.1 YANG Parser

The java parser is built with JJTree and JavaCC [3] but no external library is needed to use it.

- lexical and syntax checks are conformant to the ABNF grammar given in [1]
- semantical check covers following features :
  - name scoping and accessibility for typedef, grouping, extension, uses, leaf and leaflist, inside a module or submodule and with imported and included specifications.
  - type restriction for any type (integer, boolean, bits, float,...) and typedef
  - default value and restriction
  - augment existing node
  - Xpath for schema node in augment, leaf (of key ref type) and list (for unique statement)

2.2 Repository

jYang is an open source distribution of our toolkit under the GPL licence. The official repository is at the INRIA Gforge web site :

`http://jyang.gforge.inria.fr`

2.3 *jYang* tools

2.3.1 *jYang* parser use

jYang is distributed as a java jar file called `jyang.jar` and configured to be executable. The synoptic is :

```
java -jar jyang.jar [-h] [-f format] [-o outputfile] [-p paths] file [file]*
```

- `-h` print the synoptic
- `-f format` specifies the format for a translated output (yin format for example)
- `-o outputfile` the name of the translated output (standard output if not given) ignored if no format are given

[3] https://javacc.dev.java.net

- `-p paths` a path where to find other YANG specifications. It is needed if import or include statements are in the checked specification or if the environement variable `YANG_PATH` is not set.
- `file [file]*` specifies files containing YANG specification. It must be one specification (`module` or `submodule` for each file).

Errors Errors in YANG specifications are printed on the standard error output. *jYang* stops checking at the first lexical or syntaxical error but can find more than one semantical error. When such an error is detected, the current bloc statement is escaped and *jYang* passes to the next statement.

2.3.2 Programmatic access

jYang provides java classes and interfaces to parse YANG specification inside a java program. Internal representation of those specifications can be accessed throught the API defined in the INRIA technical report []. Below is an example of how to parse a YANG specification.

```
1  import java.io.*;
2  import jyang.*;
3
4  public class JyangTest {
5
6      /**
7      * Simple jyang test, parses and checks one YANG specification.
8      * Imported or included modules or submodules are looked in the
9      * current directory.
10     * Error messages are on the standard output
11     *
12     * @param args YANG file name
13     */
14     public static void main(String[] args) throws Exception {
15           FileInputStream yangfile = new FileInputStream(args[0]);
16           new yang(yangfile);
17           YANG_Specification spec = yang.Start();
18           spec.check();
19     }
20 }
```

The program first gets the YANG specification file at line 15. A new jyang parser is created line 16 with this file. The lexical and syntactic check are processed at line 17 and return a YANG_specification object instance that can be semantically checked, as at line 18.

2.4 Impact

The *jYang* parser and the yang API have an impact on the work of the netmod working group as they are existing implementation of a standard draft. The parser can be used by NETCONF data modelers in order to validate their models. The API allows the developpement of several backends as html or xml. *jYang* can be a support for educational purpose in master or engeneering network configuration teatching modules.

2.5 Progress Report

The java code has a size of about 15000 lines of code (without the generated code by JJTree and JavaCC) and has an executable size of 500 Ko. It was successfully tested with several YANG specifications found on YANG related site (www.netconfcentral.com and www.yang-central.org).

2.6 Conclusion

A YANG parser is achieved and an API allows YANG specification access from java programs. Current works are on the new draft version [2] and a python backend that will be integrated in the NETCONF python implementation called ENSUITE framework[5].

References

[1] M. Bjorklund. YANG - A data modeling language for NETCONF, August 2008. http://www.ietf.org/internet-drafts/draft-ietf-netmod-yang-01.txt.

[2] M. Bjorklund. YANG - A data modeling language for NETCONF, November 2008. http://www.ietf.org/internet-drafts/draft-ietf-netmod-yang-02.txt.

[3] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2). RFC 1442 (Proposed Standard), April 1993. Obsoleted by RFC 1902.

[4] J.D. Case, M. Fedor, M.L. Schoffstall, and J. Davin. Simple Network Management Protocol (SNMP). RFC 1157 (Historic), May 1990.

[5] Vincent Cridlig and Radu State. YencaP Documentation. Technical report, INRIA Lorraine, 2005.

[6] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry. The COPS (Common Open Policy Service) Protocol. RFC 2748 (Proposed Standard), January 2000. Updated by RFC 4261.

[7] R. Enns. NETCONF Configuration Protocol. RFC 4741 (Proposed Standard), December 2006.

[8] K. McCloghrie, M. Fine, J. Seligson, K. Chan, S. Hahn, R. Sahita, A. Smith, and F. Reichmeyer. Structure of Policy Provisioning Information (SPPI). RFC 3159 (Proposed Standard), August 2001.

[9] F. Strauss and J. Schoenwaelder. SMIng - Next Generation Structure of Management Information. RFC 3780 (Experimental), May 2004.