# GENERATING SKELETON CODE FOR NETCONF MODULES FROM SMI MIB MODULE DEFINITIONS

Torsten Klie
*L3S Research Center*
*Expo Plaza 1, 30539 Hannover, Germany*
klie@l3s.de

**ABSTRACT**

Configuring network devices with the Simple Network Management Protocol (SNMP) has several drawbacks. Within the IETF, a new configuration protocol is being designed: NETCONF. NETCONF offers an XML based encoding and operations to install, to manipulate, and to delete configurations of network devices. Currently there are no standard data models for NETCONF comparable to those defined in SMI MIB modules within the SNMP framework. With Yenca from Madynes (Inria, France), a modular implementation of the (not yet standardized) protocol is available. This paper presents an approach of automatic generation of skeleton code for Yenca NETCONF modules from existing SMI MIB definitions.

**KEYWORDS**

Internet Management, Configuration, NETCONF, SNMP

## 1. INTRODUCTION

In the late 1980s, the Simple Network Management Protocol (SNMP) Framework (Harrington, D. et al., 2002) started its success story, primarily due to its simplicity. It was possible to implement the protocol even on devices with limited resources as measured by then. Although SNMP advanced, there are still some open issues: one of them is configuration management. The IETF NETCONF Working Group (Leinen, S. and Bierman, A., 2005) addresses this issue. It proposes a protocol (Enns, R., 2005) that uses a data encoding based on the Extensible Markup Language (XML) (Bray, T., 2004).

In order to start using the protocol, data models for NETCONF must be designed, which can be a quite difficult task. Currently, there is a draft specification for a data model framework (Chisholm, S. and Adwankar, S., 2005) but there are no data models. For the SNMP framework, a lot of data models in form of SMI MIBs have already been specified. In order to save development time and money, it would be useful to be able to reuse these data models. This paper describes and discusses an approach for transforming SMI MIB modules to NETCONF data models in form of skeleton code for the Yenca NETCONF agent environment (State, R., 2004).

## 2. CONFIGURATION WITH SNMP

SNMP is hardly used for configuration management. Several reasons therefore can be identified. The most important ones are the following:

- SNMP is a low level technology. The protocol is simple, leaving the complexity to the management application. This is especially problematic for configuration, because configuration tasks are per se more complex than monitoring tasks.
- There is no interchange format for configuration data. Thus, if two or more agents shall receive the same configuration, the same sequence of SNMP set messages have to be sent to all the agents. There are no "`setConfig`" or "`copyConfig`" operations in SNMP.

The Structure of Management Information (SMI) (McCloghrie, K. et al., 1999) is the data modeling language corresponding to the SNMP framework. It offers simple data types that can be used for objects that are either scalars or cells within tables, i.e. two-dimensional arrays of scalars. Currently, there are more than 200 IETF Standard MIB modules defined by different IETF working groups and more than 600 vendor specific MIBs (Schönwälder, J., 2005). Although SNMP is rarely used for configuration, the MIB modules specified in the SMI language can also contain definitions of writable objects.

# 3. CONFIGURATION WITH NETCONF

## 3.1 NETCONF Protocol

The NETCONF protocol defines operations for managing network devices in a simple way. Configuration data can be retrieved, uploaded and manipulated using a formal API exposed by the managed device either as a full or a partial data set. This API is based on an XML-encoded RPC paradigm. The NETCONF architecture can be described as a four layer architecture (going from the bottom to the top):

1. Application Protocol Layer (application protocol used for transport)[1],
2. RPC Layer (Framing mechanism independent from the used application protocol),
3. Operations Layer (operations called via RPC, such as `<get-config>`, `<edit-config>`)
4. Content Layer (actual configuration data).

NETCONF agents distinguish between configuration data and state data. Data can be retrieved with the `<get-config>` operation or the `<get>` operation. The NETCONF protocol is able to deal with several sets of configuration data (data stores). For example, in addition to the active (i.e. `<running>`) configuration, there can be a `<candidate>` and a `<startup>` data store.

## 3.2 Yenca NETCONF Agent

Probably the most popular available NETCONF agent prototype is Yenca (State, R., 2004). Yenca consists of a Java-based management application and an agent written in C. The concept of the agent is modular allowing users and third parties to develop modules that can be loaded dynamically. Every module is only responsible for its core operations. Thus, when an XML request arrives, the agent parses the request, identifies the responsible module, and passes a short representation of the original request to that module. The answer to the request will be generated completely within the responsible module.

In order to plug in a new module, the author of the module has to observe certain rules:

- An arbitrary data structure for module specific data must be provided (*modulename*_ `prop_t`).
- A function for proper module registration (`register_module`) must be implemented. This function must fill out a module information data structure (`net_module_t`) which contains important information about the module (such as the module's name, a brief description, a pointer to the arbitrary data structure mentioned above, and pointers to important functions for saving and restoring configurations, data retrieval, and agent notification).
- Private functions for getting and setting the actual data must be provided.

# 4. TRANSFORMING SMI MIB MODULES TO NETCONF DATA MODELS

[1] NETCONF can use a large number of application protocols. Bindings for SSH (Wasserman, M. and Goddard, T., 2005), SOAP (Goddard, T., 2005), and BEEP (Lear, E. and Crozier, K., 2005) are already specified.

## 4.1 XML-based Data Representation

The structure of the data representation in XML is geared to the SMI to W3C XML Schema (Fallside, C. and Walmsley, P., 2004) mapping described in (Strauß, F. and Klie, T., 2003), which uses a flattened element hierarchy and does not convert the OID tree hierarchy to deeply nested elements. This improves readability a lot, compared to a straight forward mapping. However, the mapping must be slightly modified. In NETCONF, the XML data is embedded into the `<rpc-reply>` element (see Figure 1).

Figure 1. Sample NETCONF request for SNMPv2-MIB (Presuhn, R. et al., 2002)

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config><snmpv2/></get-config>
</rpc>

<rpc-reply message-id="101"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <config>
    <snmpv2><system>
        <sysContact>IBR Admins &lt;admin@ibr.cs.tu-bs.de&gt;</sysContact>
        <sysName>aligator</sysName>
        <sysLocation>TU Braunschweig, IBR, Raum 104</sysLocation>
      </system></snmpv2>
  </config>
</rpc-reply>
```

## 4.2 Separating Configuration Data from State Data

An important difference between SNMP and NETCONF is the separation between state data and configuration data. SNMP does not distinguish between these two kinds of data. Therefore, SMI does not provide any means to specify whether an object shall be considered a configuration object or state data. Thus, a simple straight-forward mapping from SMI MIB modules to NETCONF modules is not possible. How can we decide, to which kind of data an object in a MIB belongs? Separation could be based on the data types. For example, counters which have a `Counter32` or `Gauge32` data type are always state data. Objects of the `RowStatus` type can be ignored. However, there are other data types such as `OctetString` which can occur in either state, control, or configuration data.

Since it is not possible to modify them, read-only objects must be state data. Writable objects can be control or configuration data. Unfortunately, there is no possibility to distinguish automatically between control and configuration data. Therefore, annotations are needed in the SMI definitions to give the compiler a hint. For example, keywords such as `S2Y_SMI_CONFIG` can be added to the `DESCRIPTION` clauses of configuration objects. This can be done manually or with the help of an interactive script.

## 4.3 Generating Agent Code

We decided to implement the Yenca NETCONF module skeleton code generator as an smidump module. The program smidump is a MIB compiler bundled with libsmi (Strauß, F., 2004) which has a modular structure and allows to plug-in modules for different output formats (a.k.a. “output driver”).

In the following, we will explain how the generation process works using SNMPv2-MIB as an example.

1. A directory for the new module will be created. The files for the XML parser which are the same for all modules will be copied into this directory.
2. The module specific files will be generated (snmpv2-mib.h and snmpv2-mib.c in the example). The header file contain the declarations of the required functions mentioned in Section 3.2. For internal purposes, the data will be stored in a `snmpv2_prop_t` which will be defined in the header file. In the file snmpv2-mib.c, the functions declared in snmpv2-mib.h will be added. These functions can be generated automatically. Function headers for getting and setting the actual data will be added. Every object in the `snmpv2_prop_t` data structure will receive a get function and a set function. These functions cannot be generated automatically and must be implemented by the user.

3. A makefile for use with automake will be generated (Makefile.am).

At this point, the automatic conversion is finished. In order to plug the new module into the Yenca NETCONF agent, some minor modifications to the Yenca Makefiles have to be made (see the documentation shipped with Yenca (Zores, B. et al., 2004).

## 5. RELATED WORK

The draft specification of the NETCONF data model framework (Chisholm, S. and Adwankar, S., 2005) provides some general guidelines that data models for NETCONF have to observe. First of all, it requires the data modeling language to be W3C XML Schema. Every object must specify, which operations can be perfomed on it. This is done by using two mandatory appinfo elements: `<minAccess>` and `<maxAccess>` that allow for values such as `<create>`, `<delete>`, `<read>`, `<write>`, and `<execute>`. Furthermore, optional appinfo elements for status and access control are proposed.

In recent years, several smidump modules have been developed. One of them is the W3C XML Schema output driver (Strauß, F. and Klie, T., 2003) already mentioned in Section 4.1. Another output driver can be useful for the user who wants to convert MIBs: the uml output driver (Schönwälder, J. and Müller, A., 2001) which converts MIBs into reverse engineered conceptual models in form of a Dia (The GNOME Project, 2004) UML class diagrams (ISO, 2005) that helps understanding the MIB.

## 6. CONCLUSION

We have shown that it is possible to generate skeleton code for NETCONF modules using SMI MIB modules as input. Developers can reuse SMI data models with NETCONF. The transformation has been implemented for the Yenca NETCONF agent. It should be possible to generate skeleton code for other modular NETCONF agent implementations in a quite similar way.

It is questionable whether this transformation is valuable on the long run. SNMP is not used for configuration these days not only because the protocol is too low level. There is no support for complex data types in SMI which leads sometimes to complicated workarounds and hard to understand MIBs. From this point of view, completely new data models are needed. However, the approach presented in this paper can lead to a good starting point for developers who want to use NETCONF, if MIB module definitions for the objects they want to manage have been specified and implemented.

The presented approach is still work in progress. The structure of the generated code of the current implementation is quite close to the structure of the two modules that are shipped with Yenca.

Currently, there are no Schema definitions for the XML data representation. Another smidump output driver that generates W3C XML Schema definitions (or definitions in another XML schema language such as RelaxNG (Clark, J. and Makoto, M., 2001)) should be implemented. The definitions generated by the xsd output driver described in (Strauß, F. and Klie, T., 2003) are very close to the format used in our approach. Thus, that output driver could be slightly modified to generate the needed files.

## REFERENCES


Bray, T. et al., 2004. Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation.

Chisholm, S. and Adwankar, S., 2005. Framework for Netconf Data Models. Internet Draft <draft-chisholm-netconfmodel-03.txt>.

Clark, J. and Makoto, M. 2001. RELAX NG Specification. OASIS Commitee specification.

Enns, R., 2005. NETCONF Configuration Protocol. Internet Draft <draft-ietf-netconf-prot-08.txt>.

Fallside, C. and Walmsley, P., 2004. XML Schema Part 0: Primer Second Edition. W3C Recommendation.

Goddard, T., 2005. Using the Network Configuration Protocol (NETCONF) Over the Simple Object Access Protocol (SOAP). Internet Draft <draft-ietf-netconf-soap-06.txt>.

Harrington, D. et al., 2002. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC 3411.

ISO, 2005. Information technology – Open Distributed Processing – Unified Modeling Language (UML) Version 1.4.2, International Standard ISO/IEC 19501.

Lear, E. and Crozier, K., 2005. Using the NETCONF Protocol over Blocks Exchange Protocol (BEEP). Internet Draft <draft-ietf-netconf-beep-06.txt>.

Leinen, S. and Bierman, A., 2005. Network Configuration (netconf) Charter. http://www.ietf.org/html.charters/netconf-charter.html.

McCloghrie, K. et al., 1999. Structure of Management Information Version 2 (SMIv2). RFC 2578.

Presuhn, R. et al., 2002. Management Information Base (MIB) for the Simple Network Management Protocol (SNMP). RFC 3418.

Schönwälder, J., 2005. Characterization of SNMP MIB Modules. *Proceedings of 9th IFIP/IEEE International Symposium on Integrated Network Management*, Nice, France, pp. 615–628.

Schönwälder, J. and Müller, A., 2001. Reverse Engineering Internet MIBs. *Proceedings of 7th IFIP/IEEE International Symposium on Integrated Network Management*, Seattle, USA, pp. .

State, R., 2004. YENCA. WWW Page. http://sourceforge.net/projects/yenca/.

Strauß, F., 2004. Libsmi – A Library to Access SMI MIB Information. WWW Page. http://www.ibr.cs.tu-bs.de/projects/libsmi/.

Strauß, F. and Klie, T., 2003. Towards XML oriented Internet Management. In *Proc. 8th IFIP/IEEE International Symposium on Integrated Network Management*, Colorado Springs, USA, pp. 505–518.

The GNOME Project, 2004. Dia. WWW Page, http://www.gnome.org/projects/dia/

Wasserman, M. and Goddard, T., 2005. Using the NETCONF Configuration Protocol over Secure Shell (SSH). Internet Draft <draft-ietf-netconf-ssh-04.txt>.

Zores, B. et al., 2004. An Open Source Implementation of the NetConf Configuration protocol. Yenca Software Package Documentation.