SMIng:

Next Generation
Structure of Management Information
for the Internet

Frank Strauß

Department of Computer Science
TU Braunschweig
Bültenweg 74/75
38106 Braunschweig
Germany

strauss@ibr.cs.tu-bs.de
http://www.ibr.cs.tu-bs.de/˜strauss/

Problems of SMIv2

Partial Dependency on Obsolete ASN.1 Specification

- Requirements evolved over time, while SMI got stuck at 1988 ASN.1 specs.
- These ASN.1 specs got obsolete in the ISO and are hardly available.
- This results in different assumptions on some syntax details of the SMI.
- Different SMI extensions and limitations against ASN.1 make the advantages of using ASN.1 worthless.

Conclusion:

- Don't use ASN.1 or any other external specification to build a new SMI on. Instead make the new SMI self-contained by applying only common IETF standards.

Difficult to Implement Parsers

- The lack of a consistent and complete formal grammar led to erroneous SMI parser implementations written without scanner/parser generator tools.
- Therefore, many MIB modules could not be checked adequately. Many MIB modules (even IETF Standard modules) are erroneous.
- Forward references of types and objects are allowed, though hardly necessary.

Conclusion:

- Enable consistent straight-forward implementations by providing a simple but detailed formal grammar using Augmented BNF (RFC 2234).
- Choose a small set of programmer-friendly syntactical base rules.
- Enforce a fixed order of sections/statements to unify the stucture of modules.

Lack of Extensibility

- In the past, different extensions became generally necessary or wishful in certain areas. They require recycling in the standards process or must be handled independent from SMI modules.
- In some environments it may be helpful to add annotations to existing MIB definitions, e.g. notification severity levels.

Conclusion:

- Need an extension mechanism that does not break on unknown statements.
- Need a way to uniquely identify extensions used within a module.
- Tradeoff: degree of formal syntax and semantics specification.

Missing Base Types

- Some base types have been missing, are still missing, and might be missing in the future, e.g. 64-Bit integers, floating point types.

Conclusion:

- Establish a full orthogonal set of base types.

Huge Investments on SMIv2 Modules

- 100+ IETF Standard MIB modules with 10,000+ definitions.
- Even more vendor MIB modules.

Conclusion:

- Strong SMIv2 backward compatibility is a must!

SMIng Project Output

- SMIng Specification:
  an Internet-Draft maintained within the IRTF Network Management Research Group (NMRG) specifying
  - the SMIng base types,
  - the SMIng file structure including all statements,
  - three SMIng base modules,
  - the formal SMIng ABNF grammar.
- SMIng parser implementation:
  - built straight-forward from the ABNF grammar using flex/bison,
  - integrated into libsmi (discussed in the next section)

libsmi — A Library for Accessing SMI MIB Information

Current Situation

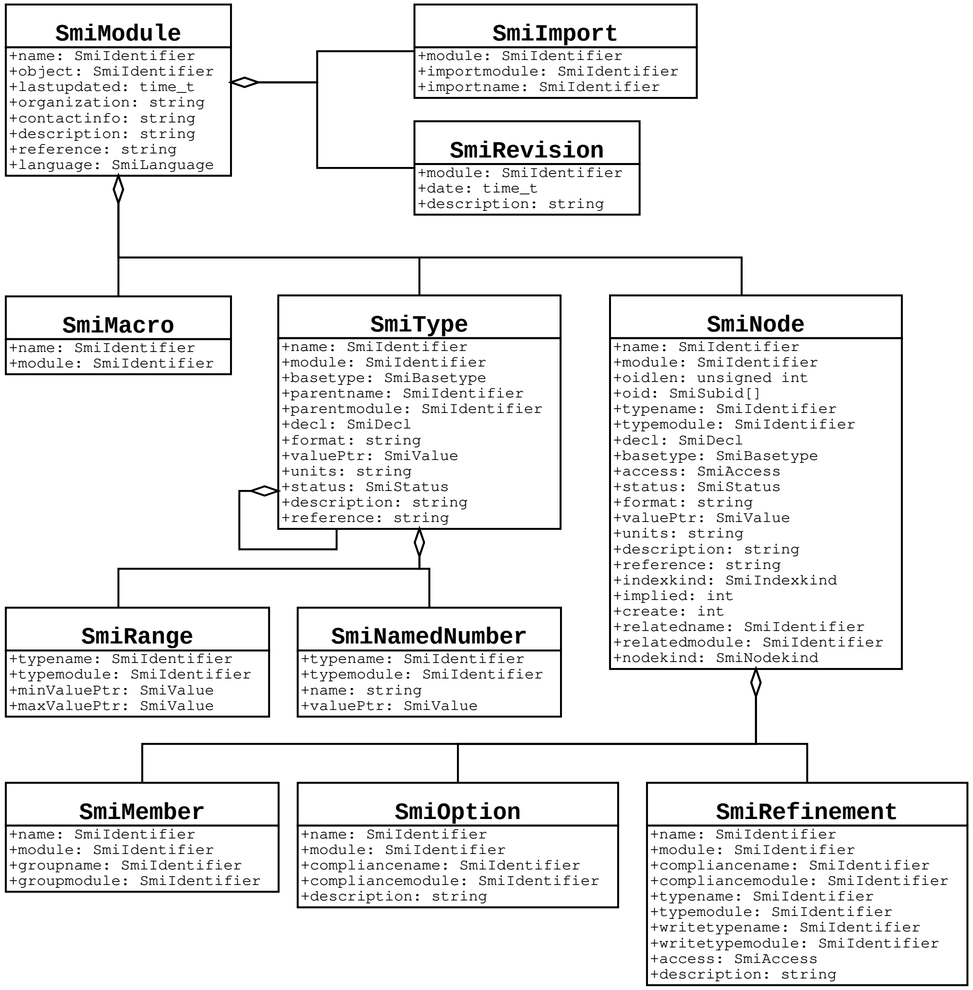
- Management applications need information from MIB modules.
- This means managing MIB module files, parsing them, and managing MIB data in memory.
- Different implementors choose different ways (with different limitations).
- This increases work to manage the same MIB modules used by multiple applications.

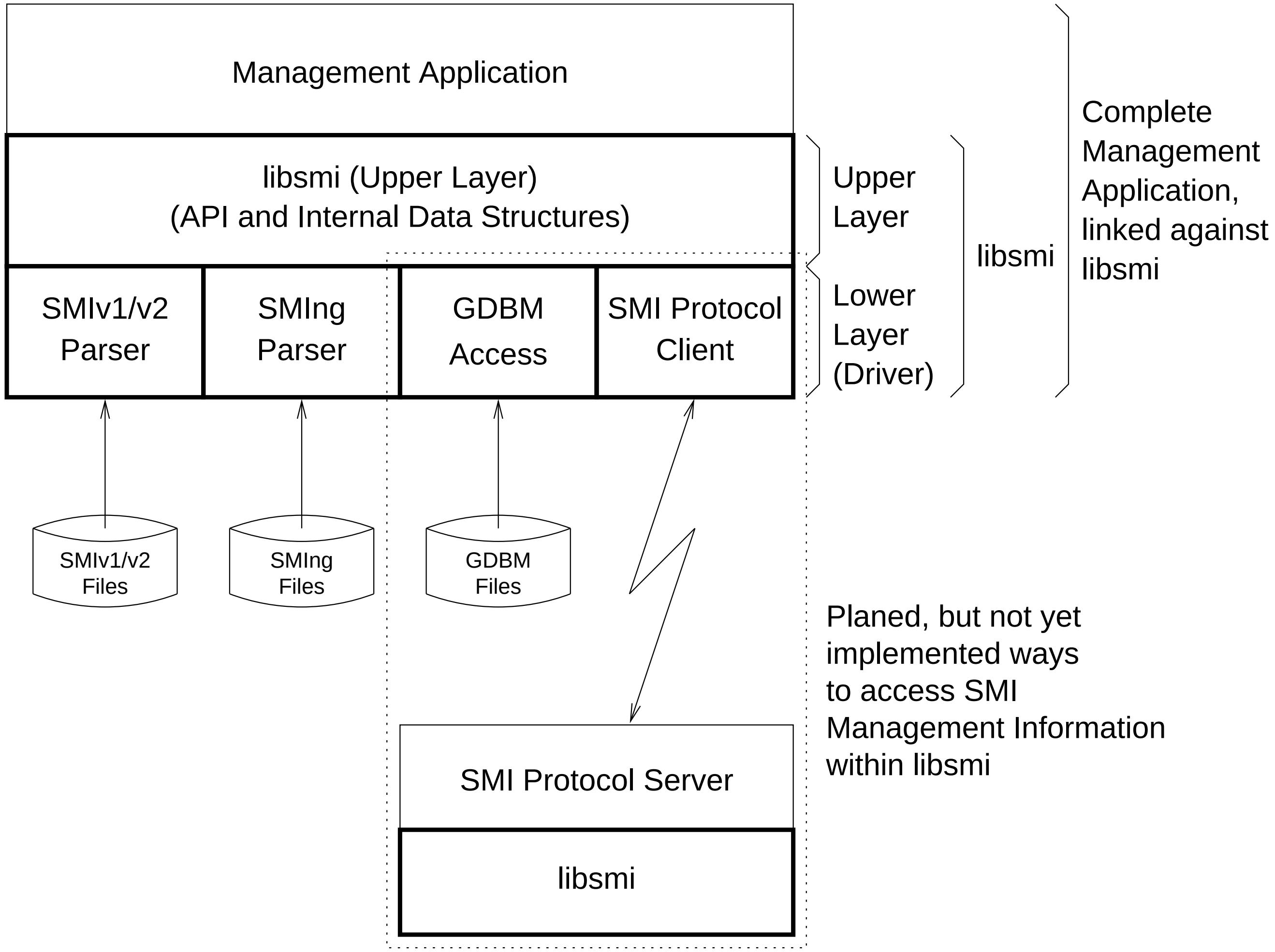
Aims

- Design an API for accessing SMI MIB Information from applications.
- Implement this API as a portable C library.

Design Overview

- The API provides MIB information independent from the underlying MIB module language (SMIv1/SMIv2/SMIng).
- The API provides
  - `smiGetXXX()` functions to retrieve MIB items by their key values,
  - `smiGetFirstXXX()` and `smiGetNextXXX()` iterator functions that walk through lists or trees of items,
  - functions to initialize and configure the behaviour of the library.
- A modular library architecture allows to support additional MIB module languages/sources in the future.


SmiModule
+name: SmiIdentifier
+object: SmiIdentifier
+lastupdated: time_t
+organization: string
+contactinfo: string
+description: string
+reference: string
+language: SmiLanguage
SmiImport
+module: SmiIdentifier
+importmodule: SmiIdentifier
+importname: SmiIdentifier
SmiRevision
+module: SmiIdentifier
+date: time_t
+description: string
SmiMacro
+name: SmiIdentifier
+module: SmiIdentifier
SmiType
+name: SmiIdentifier
+module: SmiIdentifier
+basetype: SmiBasetype
+parentname: SmiIdentifier
+parentmodule: SmiIdentifier
+decl: SmiDecl
+format: string
+valuePtr: SmiValue
+units: string
+status: SmiStatus
+description: string
+reference: string
SmiNode
+name: SmiIdentifier
+module: SmiIdentifier
+oidlen: unsigned int
+oid: SmiSubid[]
+typename: SmiIdentifier
+typemodule: SmiIdentifier
+decl: SmiDecl
+basetype: SmiBasetype
+access: SmiAccess
+status: SmiStatus
+format: string
+valuePtr: SmiValue
+units: string
+description: string
+reference: string
+indexkind: SmiIndexkind
+implied: int
+create: int
+relatedname: SmiIdentifier
+relatedmodule: SmiIdentifier
+nodekind: SmiNodekind
SmiRange
+typename: SmiIdentifier
+typemodule: SmiIdentifier
+minValuePtr: SmiValue
+maxValuePtr: SmiValue
SmiNamedNumber
+typename: SmiIdentifier
+typemodule: SmiIdentifier
+name: string
+valuePtr: SmiValue
SmiMember
+name: SmiIdentifier
+module: SmiIdentifier
+groupname: SmiIdentifier
+groupmodule: SmiIdentifier
SmiOption
+name: SmiIdentifier
+module: SmiIdentifier
+compliancename: SmiIdentifier
+compliancemodule: SmiIdentifier
+description: string
SmiRefinement
+name: SmiIdentifier
+module: SmiIdentifier
+compliancename: SmiIdentifier
+compliancemodule: SmiIdentifier
+typename: SmiIdentifier
+typemodule: SmiIdentifier
+writetypename: SmiIdentifier
+writetypemodule: SmiIdentifier
+access: SmiAccess
+description: string


Management Application
libsmi (Upper Layer)
(API and Internal Data Structures)
SMIv1/v2
Parser
SMIng
Parser
GDBM
Access
SMI Protocol
Client
SMIv1/v2
Files
SMIng
Files
GDBM
Files
SMI Protocol Server
libsmi
Upper
Layer
Lower
Layer
(Driver)
libsmi
Complete
Management
Application,
linked against
libsmi
Planed, but not yet
implemented ways
to access SMI
Management Information
within libsmi

Applications in the libsmi Package:

- MIB Checker:
  `smilint` runs a parser with increased verbosity to check MIB modules for syntax errors and inadequate constructs.
- MIB Compiler:
  `smidump` allows to dump the contents of a module in various formats:
  SMIv1, SMIv2, SMIng, Import Trees, Type Definition Trees, OID Node Trees, MOSY Style, CORBA IDL definitions according to the JIDM Specs.
- Single MIB Queries:
  `smiquery` can be used to query single items from the command line.

Other Software using libsmi:

- The GNOME SNMP Manager Project “GxSNMP”.
- An enhanced `tcpdump`.

The integration into other applications, like scotty, is planned.

References

- SMIng Specification:
  `draft-irtf-nmrg-sming-00.txt` from the IETF I-D repository,
  e.g. `ftp://ietf.org/internet-drafts/`

- libsmi Implementation and Mailinglist Information:
  `http://www.ibr.cs.tu-bs.de/projects/libsmi/`