Network Working Group
Internet-Draft
Expires: May 15, 2002

F. Strauss
TU Braunschweig

November 14, 2001

SMIng Compliance
draft-ietf-sming-compl-00

Status of this Memo

Copyright Notice

Abstract

The memo [9], which has been approved by the IESG to be published as an Informational RFC, enumerates a number of objectives that the SMIng language is expected to fulfill. This memo describes the compliance of a proposal based on former IRTF NMRG [8] work and published in [10] and companion I-Ds for each of the strived objectives. This is done by including the whole list of objectives without any changes, but adding a compliance section for each of the objectives, so that reading is as easy as possible.

Table of Contents

1. Introduction

The pure purpose of this document is to describe the compliance of the current SMIng proposal which is based on former IRTF NMRG work and published in [10], [11], [12], [13], and [14], to the objectives stated and categorized in [9].

For the purpose of readability and reference this has been done by structuring this document very closely as the objectives document. Only the following things have been changed:

- the abstract states the purpose of this document,
- this first part of the Introduction has been added to explain the structure of the document,
- to each single objective's section, a `Compliance' field has been appended, leaving the objectives itself completely unchanged,
- Section 4 explains the purpose of the `Compliance' field,
- Section 6 explains that most of this document is taken from [9] and acknowledges the work of the authors of that document, and
- the references section at the end of this document has been updated to include the documents of the discussed SMIng proposal.

Hence, the document is rather long, but readers who are familiar with the objectives document, can parse it quickly and have the original descriptions of the related objectives right at hand in the known structure.

[XXX TODO: stats of compliances. XXX]

[The rest of this section is unchanged.]

This document describes the objectives for a new data definition language that can be mapped into SNMP [1], [2] and COPS-PR [3] protocol operations. It may also be translated into SMIv2 [4], [5], [6] MIBs and SPPI [7] PIBs. Concepts such as attributes, attribute groups, methods, conventions for organization into reusable data structures, and mechanisms for representing relationships are discussed.

2. Motivation

As networking technology has evolved, a diverse set of technologies has been deployed to manage the resulting products. These vary from

Web based products, to standard management protocols and text scripts. The underlying systems to be manipulated are represented in varying ways including implicitly in the system programming, via proprietary data descriptions, or with standardized descriptions using a range of technologies including MIBs, PIBs, or LDAP schemas. The result is that management interfaces for network protocols, services, and applications such as Differentiated Services may be represented in many different, inconsistent fashions.

The SMIng working group has been chartered to define a new data definition language that will eliminate the need for a separate SMIv2 and SPPI language. That is, the new language should address the needs for the current SMIv2 and SPPI languages so that over time we can all use the new language instead.

Another motivation is to permit a more expressive and complete representation of the modeled information. Examples of additional expressiveness and completeness that are considered are the ability to formally define table existence relationships, the expression of instance creation/deletion capabilities, and the ability to define attribute groups using inheritance. These additional features are discussed in subsequent sections.

It has been recognized that the two main goals of (a) merging SMIv2/SPPI and (b) enhancing the state of art in network management data modeling can lead to conflicts. In such cases, the SMIng working group's consensus is to focus on enhancing the state of art in network management data modeling.

3. Background

The Network Management Research Group (NMRG) of the Internet Research Task Force (IRTF) has researched the issues of creating a protocol-independent data definition language that could be used by multiple protocols. Because SMIv2 and SPPI are very similar, the NMRG focused on merging these two languages, but also researched ways to abstract the objectives to produce a language that could be used for other protocols, such as LDAP and Diameter. The NMRG has published the results of their work in [8], and has submitted their specification as one proposal to consider in the development of the SMIng language.

The SMIng Working Group has accepted their submission for consideration, and to use their proposal to better understand the objectives and possible obstacles to be overcome. Where useful, the NMRG proposal has been referenced in the details below.

4. Specific Objectives for SMIng

The following sections define the objectives for the definition of a new data definition language. The objectives have been organized as follows: accepted objectives (Section 4.1), nice-to-have objectives (Section 4.2), and rejected objectives (Section 4.3). Each objective has the following information:

- Type: a field that identifies the type of objective, using one of the following values:
  - basic: considered a basic objective for SMIng and is contained in SMIv2 and/or SPPI.
  - align: supported in different ways in SMIv2 and SPPI and they must be aligned.
  - fix: considered a fix for a known problem in SMIv2 and/or SPPI.
  - new: considered a new feature.
- From: a field that defines the origin of the objective and that contains one or more of the following values:
  - SMI: exists in SMIv2.
  - SPPI: exists in SPPI.
  - NMRG: exists in the NMRG proposal, but not in SMIv2 or SPPI.
  - Charter: exists in working group charter.
  - WG: proposed during working group discussions.
- Description: a quick description of the objective.
- Motivation: rationale for the objective.
- Notes: optional notes about an objective. For example, for nice-to-have or rejected this may contain reasoning why this objective is not required by the SMIng working group, but justification why it should be considered anyway. Notes may be the opinions of the participants in the discussion on objectives and as such should not be taken as consensus of the working group or the recommendation of the objectives editing team.
- Compliance: a compliance statement that expresses whether and to which degree this objective is met by the current NMRG-based SMIng

proposal. These compliance sections are the core of this document.

4.1 Accepted Objectives

This section represents the list of objectives that have been accepted by the SMIng working group as worthwhile and therefore deserving of further consideration. Each of these objectives must be evaluated by the working group to determine if the benefit incurs an acceptable level of cost. An accepted objective may subsequently be rejected if the cost/benefit analysis determines that the benefit does not justify the cost or that the objective is in direct conflict with one or more other accepted objectives that are deemed more important.

4.1.1 The Set of Specification Documents

Type: new

From: NMRG

Description: SMIv2 is defined in three documents, based on an obsolete ITU ASN.1 specification. SPPI is defined in one document, based on SMIv2. The core of SMIng must be defined in one document and must be independent of external specifications.

Motivation: Self-containment.

Compliance: Ok. The core of the NMRG based SMIng proposal is completely defined in [10]. There are no significant dependencies on non-IETF standards like ASN.1.

4.1.2 Textual Representation

Type: basic

From: SMI, SPPI, WG

Description: SMIng definitions must be represented in a textual format.

Motivation: General IETF consensus.

Compliance: Ok. The NMRG based SMIng proposal uses a plain text format and no binary or XML representation.

4.1.3 Human Readability

Type: basic

From: WG

Description: The syntax must make it easy for humans to directly read and write SMIng modules. It must be possible for SMIng module authors to produce SMIng modules with text editing tools.

Motivation: The syntax must make it easy for humans to read and write SMIng modules.

Compliance: Ok. SMIng has a quite simple, consistently structured grammar that looks more friendly than ASN.1 and is well documented without relying on an external long-banned ITU standard or an XML representation that would bloat SMIng modules. To some degree the grammar looks similar to the well known C and Java programming languages.

4.1.4 Rigorously Defined Syntax

Type: new

From: NMRG

Description: There must be a rigorously defined syntax for the SMIng language.

Motivation: An unambiguous language promotes consistency across vendors so that different parsers produce the same results. It also provides authoritative rules to SMIng modules designers.

Compliance: Ok. The syntax is formally precisely defined by an ABNF grammar.

4.1.5 Accessibility

Type: align

From: SMI, SPPI

Description: Attribute definitions must indicate whether attributes can be read, written, created, deleted, and whether they are accessible for notifications, or are not accessible. Align PIB-ACCESS and MAX-ACCESS, and PIB-MIN-ACCESS and MIN-ACCESS.

Motivation: Alignment of SMIv2 and SPPI.

Compliance: Nearly ok. `eventonly', `readonly', and `readwrite' are supported by the core SMIng specification. The restriction to `non-accessible' and the ability of row creation/deletion are protocol specific. This has to be expressed in the SNMP and COPS-PR mappings, e.g. the SNMP mapping supports the `create' keyword. A `noaccess' keyword has to be added to the SNMP mapping.

4.1.6 Language Extensibility

Type: new

From: NMRG

Description: The language must have characteristics, so that future modules can contain information of future syntax without breaking original SMIng parsers.

E.g., when SMIv2 introduced REFERENCEs it would have been nice if it would not have broken SMIv1 parsers.

Motivation: Achieve language extensibility without breaking core compatibility.

Compliance: Ok. The core syntax is based on statement keywords, followed by an number of arbitrarily complex arguments and finished by a semicolon. This allows parsers to recognize but ignore unknown statements seamlessly. Furthermore, SMIng supports explicit language extensions. New statements can be defined along with a description and an ABNF grammar. They can be used in the module where they are defined or in other modules if they are imported. However, the language has no mechanism to define the context in which a language extension can be used.

4.1.7 Special Characters in Text

Type: new

From: WG

Description: Allow an escaping mechanism to encode special characters, e.g. double quotes and new-line characters, in text such as DESCRIPTIONs or REFERENCEs.

Motivation: ABNF can contain literal characters enclosed in double

quotes; to provide the ABNF grammar, there must be the ability to escape special characters.

Compliance: Not yet.

4.1.8 Naming

Type: basic

From: SMI, SPPI

Description: SMIng must provide mechanisms to uniquely identify attributes, groups of attributes, and events. It is necessary to specify how name collisions are handled.

Motivation: Already in SMIv2 and SPPI.

Compliance: Ok. The language has a concept of namespaces (global, module, class, named number types). Within each namespace all identifiers are required to be unique. To address items unambiguously their names can be qualified by the module name.

4.1.9 Namespace Control

Type: basic

From: SMI, SPPI

Description: There must be a hierarchical, centrally-controlled namespace for standard named items, and a distributed namespace must be supported to allow vendor-specific naming and to assure unique module names across vendors and organizations.

Motivation: Need to unambiguously identify definitions of various kinds. Some SMI implementations have problems with different objects from multiple modules but with the same name. Furthermore, the probability of module name clashes rises over time (for example, different vendors defining their own SYSTEM-MIB).

Notes: An example naming scheme is the one employed by the Java programming language with a central naming authority assigning the top-level names.

Compliance: Not yet. As with SMIv2, distributed namespaces are given by modules. However, there is no centrally controlled namespace

for module names. The current idea is to state some guidelines on module naming, e.g. encourage MIB authors to use module name prefixes that consist of vendor names or enterprise numbers. Another approach would be to add DNS domain names to module names. All these approaches would be applicable to the SMIng proposal.

4.1.10 Modules

Type: basic

From: SMI, SPPI

Description: SMIng must provide a mechanism for uniquely identifying a module, and specifying the status, contact person, revision information, and the purpose of a module.

SMIng must provide mechanisms to group definitions into modules and it must provide rules for referencing definitions from other modules.

Motivation: Modularity and independent advancement of documents.

Notes: Text about module conformance has been moved to Section 4.1.11.

Compliance: Ok. Definitions are grouped in modules. Modules are identified by names and contain status, contact, and revision clauses. The language supports explicit imports from other modules.

4.1.11 Module Conformance

Type: basic

From: SMI, SPPI

Description: SMIng must provide mechanisms to detail the minimum requirements implementers must meet to claim conformance to a standard based on the module.

Motivation: Ability to convey conformance requirements.

Compliance: Not yet. Module conformance requirements affect both core SMIng modules and mappings to SNMP and COPS-PR. Currently, there is just some first plain mapping from SMIv2 module conformance clauses to SNMP mapping specific `group' and

`compliance' clauses.

4.1.12 Arbitrary Unambiguous Identities

Type: basic

From: SMI

Description: SMI allows the use of OBJECT-IDENTITIES to define unambiguous identities without the need of a central registry. SMI uses OIDs to represent values that represent references to such identities. SMIng needs a similar mechanism (a statement to register identities, and a base type to represent values).

Motivation: SMI Compatibility.

Notes: This is an obvious objective. Additionally, everything not on the wire, such as modules, will still be assigned OIDs.

It is yet to be determined whether the assignment of the OID occurs within the core or within a protocol-specific mapping.

Compliance: Ok. The language supports an `identity' clause. In SNMP and COPS-PR mappings these identities can be mapped to registered OIDs.

4.1.13 Protocol Independence

Type: basic

From: Charter

Description: SMIng must define data definitions in support of the SNMP and COPS-PR protocols. SMIng may define data definitions in support of other protocols.

Motivation: So data definitions may be used with multiple protocols and multiple versions of those protocols.

Compliance: Ok. The core language is used to define types and data structures/classes. Protocol mappings, that can be separated from the core definition modules, are used to define the mappings of core data structures/classes to protocols like SNMP and COPS-PR (and maybe others).

4.1.14 Protocol Mapping

Type: basic

From: Charter

Description: The SMIng working group, in accordance with the working group charter, will define mappings of protocol independent data definitions to protocols based upon installed implementations. The SMIng working group can define mappings to other protocols as long as this does not impede the progress on other objectives.

Motivation: SMIng working group charter.

Compliance: Ok. There are I-Ds for SNMP and COPS-PR mappings [11], [12]. Other mappings are not subject to the current WG efforts.

4.1.15 Translation to Other Data Definition Languages

Type: basic

From: Charter

Description: SMIng language constructs must, wherever possible, be translatable to SMIv2 and SPPI. At the time of standardization of a SMIng language, existing SMIv2 MIBs and SPPI PIBs on the standards track will not be required to be translated to the SMIng language. New MIBs/PIBs will be defined using the SMIng language.

Motivation: Provide best-effort backwards compatibility for existing tools while not placing an unnecessary burden on MIBs/PIBs that are already on the standards track.

Compliance: Ok. SMIng modules per se are not protocol specific, thus they cannot be translated. However, an SMIng mapping of an SMIng core module to SNMP or COPS-PR along with the mapped SMIng module can be translated to SMIv2 or SPPI.

4.1.16 Base Data Types

Type: basic

From: SMI, SPPI

Description: SMIng must support the base data types Integer32, Unsigned32, Integer64, Unsigned64, Enumeration, Bits, OctetString,

and OID.

Motivation: Most are already common. Unsigned64 and Integer64 are in SPPI, must fix in SMI. Note that Counter and Gauge types can be regarded as derived types instead of base types.

Compliance: Ok. All the mentioned base data types are supported by the language.

4.1.17 Enumerations

Type: basic

From: SMI, SPPI

Description: SMIng must provide support for enumerations. Enumerated values must be a part of the enumeration definition.

Motivation: SMIv2 already has enumerated numbers.

Notes: Enumerations have the implicit constraint that the attribute is constrained to the values for the enumeration.

Compliance: Ok. There is an `Enumeration' base data type.

4.1.18 Discriminated Unions

Type: new

From: WG

Description: SMIng must support discriminated unions.

Motivation: Allows to group related attributes together, such as InetAddressType (discriminator) and InetAddress, InetAddressIPv4, InetAddressIPv6 (union). The lack of discriminated unions has also lead to relatively complex sparse table work-around in some DISMAN mid-level manager MIBs.

Notes: Discriminated unions have the property that the union attribute type is constrained by the value of the discriminator attribute.

Compliance: Not yet. There is just a rough idea that could look like this:

```
typedef InetAddressType {
    type  Enumeration (unknown(0), ipv4(1), ipv6(2), dns(16));
    ...
};

class InetNetworkEndpoint {
    attribute InetAddressType type { ... };
    attribute InetAddress address {
        typemap type {
            map ipv4 InetAddressIPv4;
            map ipv6 InetAddressIPv6;
            map dns  InetAddressDNS;
        };
        ...
    };
};
```

4.1.19 Instance Pointers

Type: basic

From: SMI, SPPI

Description: SMIng must allow specifying pointers to instances (i.e., a pointer to a particular attribute in a row).

Motivation: It is common practice in MIBs and PIBs to point to other instances.

Compliance: Not yet. The ObjectIdentifier base data type can be used for this purpose, but it's unclear whether we will need a mechanism to restrict instance pointer values or whether we will need a more protocol independent way to specify instance pointers.

4.1.20 Row Pointers

Type: align

From: SMI, SPPI

Description: SMIng must allow specifying pointers to rows.

Motivation: It is common practice in MIBs and PIBs to point to other rows (see RowPointer, PIB-REFERENCES).

Compliance: Halfway. The language supports a `Pointer' base data

type, which can be restricted to point to instances of a specific class and its sub-classes. The mapping of these abstract pointers to protocols is not yet done. The COPS-PR mapping seems to be fairly easy, since class instances in COPS-PR can only be identified by a single integer value. The mapping to SNMP is expected to be much more complex.

4.1.21 Constraints on Pointers

Type: align

From: SPPI

Description: SMIng must allow specifying the types of objects to which a pointer may point.

Motivation: Allows code generators to detect and reject illegal pointers automatically. Can also be used to automatically generate more reasonable implementation-specific data structures.

Notes: Pointer constraints are a special case of attribute value constraints (Section 4.3.2) in which the prefix of the OID (row or instance pointer) value is limited to be only from a particular table.

Compliance: Ok. The `Pointer' base data type can be restricted so that a pointer value can only reference instances of a specific class or its sub-classes. See also Section 4.1.20.

4.1.22 Base Type Set

Type: basic

From: SMI, SPPI

Description: SMIng must support a fixed set of base types of fixed size and precision. The list of base types must not be extensible unless the SMI itself changes.

Motivation: Interoperability.

Compliance: Ok. The language has a complete set of base data types as far as we can tell today, including 64-bit signed and unsigned integers. Even floating point types are supported, so that hopefully the need for new base data types will not arise in the foreseeable future.

4.1.23 Extended Data Types

Type: align

From: SMI, SPPI

Description: SMIng must support a mechanism to derive new types, which provide additional semantics (e.g., Counters, Gauges, Strings, etc.), from base types. It may be desirable to also allow the derivation of new types from derived types. New types must be as restrictive or more restrictive than the types that they are specializing.

Motivation: SMI uses application types and textual conventions. SPPI uses derived types.

Compliance: The language allows to define new types derived from existing defined types or base data types through the use of the `typedef' statement.

4.1.24 Units, Formats, and Default Values of Defined Types and Attributes

Type: fix

From: NMRG

Description: In SMIv2 OBJECT-TYPE definitions may contain UNITS and DEFVAL clauses and TEXTUAL-CONVENTIONs may contain DISPLAY-HINTs. In a similar fashion units and default values must be applicable to defined types and format information must be applicable to attributes.

Motivation: Some MIBs introduce TCs such as KBytes and every usage of the TC then specifies the UNITS "KBytes". It would simplify things if the UNITS were attached to the type definition itself.

Notes: The SMIng WG must clarify the behavior if an attribute uses a defined type and both, the attribute and the defined type, have units/default/format information.

Compliance: Ok. The `units' and `default' clauses are applicable to both defined types and attributes.

4.1.25 Table Existence Relationships

Type: align

From: SMI, SPPI

Description: SMIng must support INDEX, AUGMENTS, and EXTENDS in the SNMP/COPS-PR protocol mappings.

Motivation: These three table existence relationships exist either in the SMIv2 or the SPPI.

Compliance: Ok. These table indexing schemes are supported in SNMP and COPS-PR mappings.

4.1.26 Table Existence Relationships (2)

Type: new

From: NMRG

Description: SMIng must support EXPANDS and REORDERS relationships in the SNMP/COPS-PR protocol mappings.

Motivation: A REORDERS statement allows indexing orders to be swapped. An EXPANDS statement formally states that there is a 1:n existence relationship between table rows.

Compliance: Ok. These table indexing schemes are supported in SNMP mappings. For COPS-PR they are not applicable due to the simple indexing of PRCs.

4.1.27 Attribute Groups

Type: new

From: NMRG

Description: An attribute group is a named, reusable set of attributes that are meaningful together. It can be reused as the type of attributes in other attribute groups (see also Section 4.1.28). This is similar to `structs' in C.

Motivation: Required to map the same grouping of attributes into SNMP and COPS-PR tables. Allows to do index reordering without having to redefine the attribute group. Allows to group related

attributes together (e.g. InetAddressType, InetAddress).

The ability to group attributes provides an indication that the attributes are meaningful together.

Compliance: Ok. This is done by using `classes'.

4.1.28 Containment

Type: new

From: NMRG

Description: SMIng must provide support for the creation of new attribute groups from attributes of more basic types and potentially other attribute groups.

Motivation: Simplifies the reuse of attribute groups such as InetAddressType and InetAddress pairs.

Notes: Containment has the implicit existence constraint that if an instance of a contained attribute group exists, then the corresponding instance of the containing attribute group must also exist.

Compliance: Ok. Classes contain `attributes'. Each attribute has a type which can be a simple type (a base data type, or a derived type) or another class.

4.1.29 Single Inheritance

Type: new

From: NMRG

Description: SMIng must provide support for mechanisms to extend attribute groups through single inheritance.

Motivation: Allows to extend attribute groups, like a generic DiffServ scheduler, with attributes for a specific scheduler, without cut&paste.

Notes: Single inheritance with multiple levels (e.g., C derives from B, and B derives from A) must be allowed.

Inheritance has the implicit existence constraint that if an

instance of a derived attribute group exists, then the corresponding instance of the base attribute group must also exist.

Inheritance could help to add attributes to an attribute group that are specific to a certain protocol mapping and do not appear in the protocol-neutral attribute group.

Compliance: Ok. Classes can be derived from parent classes.

4.1.30 Reusable vs. Final Attribute Groups

Type: new

From: NMRG, WG

Description: SMIng must differentiate between "final" and reusable attribute groups, where the reuse of attribute groups covers inheritance and containment.

Motivation: This information gives people more information how attribute groups can and should be used. It hinders them from misusing them.

Notes: This objective attempts to convey the idea that some attribute groups are not meant to stand on their own and instead only make sense if contained within another attribute group.

Compliance: Unclear. Currently, there is some special semantic defined for the `unique' statement of a class: if it's missing, the class is reusable; if it's present, the class is final. However, it's unclear at this point in time whether this is a wise language design.

4.1.31 Events

Type: basic

From: SMI, SPPI

Description: SMIng must provide mechanisms to define events which identify significant state changes.

Motivation: These represent the protocol-independent events that lead to SMI notifications or SPPI reports.

Compliance: Ok. The language allows to define `events'. Note that events are bound to classes, i.e. an event instance is implicitly related to a class instance.

4.1.32 Creation/Deletion

Type: align

From: SMI, SPPI

Description: SMIng must support a mechanism to define creation/deletion operations for instances. Specific creation/deletion errors, such as INSTALL-ERRORS, must be supported.

Motivation: Available for row creation in SMI, and available in SPPI.

Compliance: Ok. The SNMP mapping supports the `create' keyword to denote that rows within a table can be created and deleted through SNMP protocol operations. COPS-PR supports specific create and delete protocol operations. Explicit statements that these operations are applicable to a PRC don't seem to be required.

4.1.33 Range and Size Constraints

Type: basic

From: SMI, SPPI

Description: SMIng must allow specifying range and size constraints where applicable.

Motivation: The SMI and SPPI both support range and size constraints.

Compliance: Ok. Range and size constraints are supported.

4.1.34 Uniqueness

Type: basic

From: SPPI

Description: SMIng must allow the specification of uniqueness constraints on attributes. SMIng must allow the specification of multiple independent uniqueness constraints.

Motivation: Knowledge of the uniqueness constraints on attributes allows to verify protocol specific mappings (e.g. INDEX clauses). The knowledge can also be used by code generators to improve generated implementation-specific data structures.

Compliance: Ok. There's a `unique' statement. It can be applied multiple times to a common class.

4.1.35 Extension Rules

Type: basic

From: SMI, SPPI

Description: SMIng must provide clear rules how one can extend SMIng modules without causing interoperability problems "over the wire".

Motivation: SMIv2 and SPPI have extension rules.

Compliance: Not yet. There are some initial rules, but they must be refined very carefully especially due to class inheritence and composition complications.

4.1.36 Deprecate Use of IMPLIED Keyword

Type: fix

From: WG

Description: The SMIng SNMP mapping must deprecate the use of the IMPLIED indexing schema.

Motivation: IMPLIED is confusing and most people don't understand it. The solution (IMPLIED) is worse than the problem it is trying to solve and therefore for the sake of simplicity, the use of IMPLIED should be deprecated.

Compliance: Nearly ok. The core language does not care about index encoding at all. The SNMP still supports an `implied' keyword just to retain SMIv2 compatibility. The wording has to be changed to make clear that future SNMP mappings MUST NOT use the `implied' keyword.

4.1.37 No Redundancy

Type: fix

From: NMRG

Description: The SMIng language must avoid redundancy.

Motivation: Remove any textual redundancy for things like table entries and SEQUENCE definitions, which only increase specifications without providing any value.

Compliance: Ok. In the core language and mappings some unnecessary redundant definition are no longer present, e.g. table columns are now expressed by nesting attributes (columns) within a class (table).

4.1.38 Compliance and Conformance

Type: basic

From: SMI, SPPI

Description: SMIng must provide a mechanism for compliance and conformance specifications for protocol-independent definitions as well as for protocol mappings.

Motivation: This capability exists in SMIv2 and SPPI. The NMRG proposal has the ability to express much of this information at the protocol-dependent layer. Some compliance or conformance information may be protocol-independent, therefore there is also a need to be able to express this information protocol-independent part.

Compliance: Not yet. Currently, only the SNMP mapping supports something similar to the SMIv2 conformance statements.

4.1.39 Allow Refinement of All Definitions in Conformance Statements

Type: fix

From: WG

Description: SMIv2, RFC 2580, Section 3.1 says:

> The OBJECTS clause, which must be present, is used to specify each object contained in the conformance group. Each of the specified objects must be defined in the same information module as the OBJECT-GROUP macro appears, and must have a MAX-ACCESS clause value of "accessible-for-notify", "read-only", "read-write", or "read-create".

The last sentence forbids to put a not-accessible INDEX object into an OBJECT-GROUP. Hence, you can not refine its syntax in a compliance definition. For more details, see http://www.ibr.cs.tu-bs.de/ietf/smi-errata/

Motivation: This error should not be repeated in SMIng.

Compliance: Not yet.

4.1.40 Categories

Type: basic

From: SPPI

Description: SMIng must provide a mechanism to group definitions into subject categories. Concrete instances may only exist in the scope of a given subject category or context.

Motivation: To scope the categories to which a module applies. In SPPI this is used to allow a division of labor between multiple client types.

Compliance: Ok. The core language and the SNMP mapping do not require subject categories. The COPS-PR mapping support the `clienttype' statement.

4.1.41 Core Language Keywords vs. Defined Identifiers

Type: fix

From: NMRG

Description: In SMI and SPPI modules some language keywords (macros and a number of basetypes) have to be imported from different SMI language defining modules, e.g. OBJECT-TYPE, MODULE-IDENTITY, Integer32 must to be imported from SNMPv2-SMI and TEXTUAL-CONVENTION must be imported from SNMPv2-TC, if used. MIB authors

are continuously confused about these import rules. In SMIng only defined identifiers must be imported. All SMIng language keywords must be implicitly known and there must not be a need to import them from any module.

Motivation: Reduce confusion. Clarify the set of language keywords.

Compliance: Ok. The language is clearly defined: All defined non-local identifiers MUST be imported. All language keywords MUST NOT be imported.

4.1.42 Instance Naming

Type: align

From: SMI, SPPI

Description: Instance naming in SMIv2 and SPPI is different. SMIng must align the instance naming (either in the protocol neutral model or the protocol mappings).

Motivation: COPS-PR and SNMP have different instance identification schemes that must be handled.

Notes: A solution requires to investigate how close the naming schemes dictated by the protocols are. Perhaps it is feasible to have a single instance naming scheme in both SNMP and COPS-PR, even though the current SPPI and SMIv2 are different.

Compliance: Ok. COPS-PR and SPPI are Proposed Standards. It seems to be impossible to change such a fundamental thing like the naming scheme in SPPI. Hence, the SMIng framework has to handle the different naming schemes. This is supported by separating the SNMP and COPS-PR protocol mappings from the core definitions. Only the protocol mappings support the appropriate instance naming scheme.

4.1.43 Length of Identifiers

Type: fix

From: NMRG

Description: The allowed length of the various kinds of identifiers must be extended from the current `should not exceed 32' (maybe even from the `must not exceed 64') rule.

Motivation: Reflect current practice of definitions.

Notes: The 32-rule was added back in the days where compilers could not deal with long identifiers. This rule is continuously violated these days and it does not make sense to keep it.

Compliance: Ok. The language only has a `must not exceed 64' rule.

4.1.44 Assign OIDs in the Protocol Mappings

Type: new

From: NMRG

Description: SMIng must not assign OIDs to reusable definition of attributes, attribute groups, events, etc. Instead, SNMP and COPS-PR mappings must assign OIDs to the mapped items.

Motivation: Assignment of OIDs in protocol neutral definitions can complicate reuse. OIDs of synonymous attributes are not the same in SMI and SPPI definitions. MIBs and PIBs are already registered in different parts of the OID namespace.

Compliance: Ok. OID assignment are only done in the protocol mappings.

4.2 Nice-to-Have Objectives

This section represents the list of recommended objectives that would be nice to have. However, these are not automatically thought of as accepted objectives as, for example, they may entail a non-trivial amount of work in underlying protocols to support or they may be regarded as less important than other contradicting objectives that are accepted.

4.2.1 Methods

Type: new

From: WG

Description: SMIng should support a mechanism to define method signatures (parameters, return values, exception) that are implemented on agents.

Motivation: Methods are needed to support the definition of

operational interfaces such as found in [RFC2925] (ping, traceroute and lookup operations). Also, the ability to define constructor/destructor interfaces could address issues such as encountered with SNMP's RowStatus solution.

Notes: Is it possible to do methods without changing the underlying protocol? There is agreement that methods are useful, but disagreement upon the impact - one end of the spectrum sees this as a documentation tool for existing SNMP capabilities, while the other end sees this as a protocol update, moving forward, to natively support methods. The proposal is to wait and see if this is practical to implement as a syntax that is useful and can map to the protocol.

Compliance: No. We probably don't want to support methods.

4.2.2 Unions

Type: new

From: WG

Description: SMIng should support a standard format for unions.

Motivation: Allows an attribute to contain one of many types of values. The lack of unions has also lead to relatively complex sparse table work-around in some DISMAN mid-level managers. Despite from discriminated unions (see Section 4.1.18), this kind of union has no accompanied explicit discriminator attribute that selects the union's type of value.

Notes: The thought is that SNMP and COPS-PR can already support unions because they do not care about what data type goes with a particular OID.

Compliance: No. However, there is an idea for a more specific kind of unions, discriminated unions, which are expected to be more useful, so that we probably don't want to support (non-discriminated) unions. See also Section 4.1.18.

4.2.3 Float Data Types

Type: new

From: WG, NMRG

Description: SMIng should support the base data types Float32, Float64, Float128.

Motivation: Missing base types can hurt later on, because they cannot be added without changing the language, even as an SMIng extension. Lesson learnt from the SMIv1/v2 debate about Counter64/Integer64/...

Notes: There is no mention as to whether or not the underlying protocols will have to natively support float data types. This is left to the mapping. However, it seems imperative that the float data type needs to be added to the set of intrinsic types in the SMIng language at the creation of the language as it will be impossible to add them later without changing the language.

Compliance: Ok. Float32, Float64, and Float128 are supported. They represent standard IEEE 754 floating point types.

4.2.4 Comments

Type: fix

From: NMRG

Description: The syntax of comments should be well defined, unambiguous and intuitive to most people, e.g., the C++/Java `//' syntax.

Motivation: ASN.1 Comments (and thus SMI and SPPI comments) have been a constant source of confusion. People use arbitrary lengthy strings of dashes (`----------') in the wrong assumption that this is always treated as a comment. Some implementations try to accept these syntactically wrong constructs which even raises confusion. We should get rid of this problem.

Notes: If the SMIng working group adopts a C-like syntax, then the C++/Java single-line comment should be adopted as well.

Compliance: Ok. Comments start at `//' and end at the end of line.

4.2.5 Referencing Tagged Rows

Type: align

From: SPPI

Description: PIB and MIB row attributes reference a group of entries in another table. SPPI formalizes this by introducing PIB-TAG and PIB-REFERENCES clauses. This functionality should be retained in SMIng.

Motivation: SPPI formalizes tag references. Some MIBs also use tag references (see SNMP-TARGET-MIB in RFC2573) even though SMIv2 does not provide a formal notation.

Compliance: No. It's unclear whether and how to address this issue.

4.2.6 Arrays

Type: new

From: WG

Description: SMIng should allow the definition of a SEQUENCE OF attributes or attribute groups (Section 4.1.27).

Motivation: The desire for the ability to have variable-length, multi-valued objects.

Notes: Some issues with arrays are still unclear. As long as there are no concepts to solve the problems with access semantics (how to achieve atomic access to arbitrary-sized arrays) and their mappings to SNMP and COPS-PR protocol operations, arrays cannot be more than a nice to have objective.

Compliance: No. Since arrays access semantics are unclear, arrays are not supported. In case arrays would be just another approach to define tables, we would prefer to prevent redundant ways to define the same thing.

4.2.7 Internationalization

Type: new

From: WG

Description: Informational text (DESCRIPTION, REFERENCE, ...) should allow i18nized encoding, probably UTF-8.

Motivation: There has been some demand for i18n in the past. The BCP RFC 2277 demands for internationalization.

Notes: Although English is the language of IETF documents, SMIng should allow other languages for private use.

Compliance: No. But of course, it's easy to add appropriate words. This will be done with the next revision.

4.2.8 Separate Data Modeling from Management Protocol Mapping

Type: new

From: NMRG

Description: It should be possible to separate the domain specific data modeling work from the network management protocol specific work.

Motivation: Today, working groups designing new protocols are forced to care about the design of SNMP MIBs and maybe COPR-PR PIBs to manage the new protocol. This means that experts in a specific domain are faced with details of at least one foreign (network management) technology. This leads to hard work and long revision processes. It would be a win to separate the task of pure data modeling which can be done by the domain experts easily from the network management protocol specific mappings. The mapping to SNMP and/or COPS-PR can be done (a) later separately and (b) by network management experts. This required NM expertise no longer hinders the progress of the domain specific working groups.

Compliance: Ok. This is done by the facility to define core SMIng types and classes in one module, and SNMP and COPS-PR mappings in other modules. However, it's also possible to combine them in a single module.

4.3 Rejected Objectives

This section represents the list of objectives that were rejected during the discussion on the objectives. Those objectives that have been rejected need not be addressed by SMIng. This does not imply that they must not be addressed.

4.3.1 Incomplete Translations

Type: basic

From: WG

Description: Reality sucks. All information expressed in SMIng may not be directly translatable to a MIB or PIB construct, but all information should be able to be conveyed in documentation or via other mechanisms.

Motivation: SMIng working group requires this to ease transition.

Notes: The SMIng language itself cannot require what compilers do that translate SMIng into something else. So this seems to fall out of the scope of the current working group charter.

Compliance: No. This is out of scope of the SMIng specification. It's an implementation issue.

4.3.2 Attribute Value Constraints

Type: new

From: WG

Description: SMIng should provide mechanisms to formally specify constraints between values of multiple attributes.

Motivation: Constraints on attribute values occur where one or more attributes may affect the value or range of values for another attribute. One such relationship exists in IPsec, where the type of security algorithm determines the range of possible values for other attributes such as the corresponding key size.

Notes: This objective as is has been rejected as too general, and therefore virtually impossible to implement. However, constraints that are implicit with discriminated unions (Section 4.1.18), enumerated types (Section 4.1.17), pointer constraints (Section 4.1.21)), etc., are accepted and these implicit constraints are mentioned in the respective objectives.

Compliance: No. This is too general and out of scope, but SMIng extensions may be defined in the future to support this.

4.3.3 Attribute Transaction Constraints

Type: new

From: WG

Description: SMIng should provide a mechanism to formally express

that certain sets of attributes can only be modified in combination.

Motivation: COPS-PR always does operations on table rows in a single transaction. There are SMIv2 attribute combinations that need to be modified together (such as InetAddressType, InetAddress).

Notes: Alternative is to either use Methods (Section 4.2.1) or assume that all attributes in an attribute group (Section 4.1.27) are to be considered atomic.

Compliance: No. Again, this is out of scope, but SMIng extensions may be defined in the future to support this.

4.3.4 Method Constraints

Type: new

From: WG

Description: Method definitions should provide constraints on parameters.

Motivation: None.

Notes: Unless methods (Section 4.2.1) are done, there is no use for this. Furthermore, this objective has not been motivated by any proponent.

Compliance: No. Methods are not supported at all.

4.3.5 Agent Capabilities

Type: basic

From: SMI

Description: SMIng should provide mechanisms to describe agent implementations.

Motivation: To permit manager to determine variations from the standard for an implementation.

Notes: Agent capabilities should not be part of SMIng, but should instead be a separate capabilities table.

Compliance: Ok. Agent capabilities have very rarely been used in the past and seem to be more specification overhead than useful. Hence, agent capabilities are not part of the mandatory language core. However, [11] specifies an `agentcaps' language extension that allows to specify agent capabilities.

4.3.6 Relationships

Type: new

From: NMRG, WG

Description: Ability to formally depict existence dependency, value dependency, aggregation, containment, and other relationships between attributes or attribute groups.

Motivation: Helps humans to understand the conceptual model of a module. Helps implementers of MIB compilers to generate more `intelligent' code.

Notes: This objective was deemed too general to be useful and instead the individual types of relationship objectives (e.g., pointers, inheritance, containment, etc.) are evaluated on a case-by-case basis with the specific relationships deemed useful being included as accepted objectives.

Compliance: No. This is out of scope, but SMIng extensions may be defined in the future to support this.

4.3.7 Procedures

Type: new

From: WG

Description: SMIng should support a mechanism to formally define procedures that are used by managers when interacting with an agent.

Motivation: None.

Notes: This objective has not been motivated by any proponent.

Compliance: No. This is out of scope, but SMIng extensions may be defined in the future to support this.

4.3.8 Associations

Type: new

From: WG

Description: SMIng should provide mechanisms to explicitly specify associations.

Motivation: None.

Notes: This objective has not been motivated by any proponent.

Compliance: No. This is out of scope, but SMIng extensions may be defined in the future to support this.

4.3.9 Association Cardinalities

Type: new

From: WG

Description: Cardinalities between associations should be formally defined.

Motivation: If you have an association between attribute groups A and B, the cardinality of A indicates how many instances of A may be associated with a single instance of B. Our discussions in Minneapolis indicated that we want to convey "how many" instances are associated in order to define the best mapping algorithm - whether a new table, a single pointer, etc. For example, do we use RowPointer or an integer index into another table? Do we map to a table that holds instances of the association/relationship itself?

Notes: Without associations (Section 4.3.8), this has no use.

Compliance: No. This is out of scope, but SMIng extensions may be defined in the future to support this.

4.3.10 Categories of Modules

Type: new

From: WG

Description: The SMIng documents should give clear guidance on which kind of information (with respect to generality, type/attribute group/extension/..) should be put in which kind of a module.

E.g., in SMIv2 we don't like to import Utf8String from SYSAPPL-MIB, but we also do not like to introduce a redundant definition.

A module review process should probably be described that ensures that generally useful definitions do not go into device or service specific modules.

Motivation: Bad experience with SMIv2.

Notes: It is not clear how this can be done with the language to be created by SMIng WG.

Compliance: Not yet. There are plans to create a guidelines document for SMIng module authors that contains useful information that does not fit into the formal language specification.

4.3.11 Mapping Modules to Files

Type: new

From: NMRG

Description: There should be a clear statement how SMIng modules are mapped to files (1:1, n:1?) and how files should be named (by module name in case of 1:1 mapping?).

Motivation: SMI implementations show up a variety of filename extensions (.txt, .smi, .my, none). Some expect all modules in a single file, others don't. This makes it more difficult to exchange modules.

Notes: This is just an implementation detail and is best left to a BCP and not made a part of the language definition.

Compliance: No. This is an implementation issue.

4.3.12 Simple Grammar

Type: new

From: NMRG

Description: The grammar of the language should be as simple as possible. It should be free of exception rules. A measurement of simplicity is shortness of the ABNF grammar.

Motivation: Ease of implementation. Ease of learning/understanding.

Notes: This seems like an obvious objective, however shortness of the ABNF grammar is not necessarily a reflection of the simplicity of the grammar.

Compliance: Ok. The grammar follows some simple rules without any exceptions.

4.3.13 Place of Module Information

Type: fix

From: NMRG

Description: Module specific information (organization, contact, description, revision information) should be bound to the module itself and not to an artificial node (like SMIv2 MODULE-IDENTITY).

Motivation: Simplicity and design cleanup.

Notes: This does not seem to be a problem with the current SMI. Although simplification is a good thing, this detail is not considered an objective.

Compliance: Ok. Module contact, description and revision information is contained immediately in the module body, instead of a sub-statement of the module.

4.3.14 Module Namespace

Type: new

From: WG

Description: Currently the namespace of modules is flat and there is no structure in module naming causing the potential risk of name clashes. Possible solutions:

* Assume module names are globally unique (just as SMIv1/v2), just give some recommendations on module names.

* Force all organizations, WGs and vendors to apply a name prefix (e.g. CISCO-GAGA-MIB, IETF-DISMAN-SCRIPT-MIB?).

* Force enterprises to apply a prefix based on the enterprise number (e.g. ENT2021-SOME-MIB).

* Put module names in a hierarchical domain based namespace (e.g. DISMAN-SCRIPT-MIB.ietf.org).

Motivation: Reduce risk of module name clashes.

Notes: Some aspects of this objective overlap with other objectives (namespace control (Section 4.1.9)) and other aspects were thought best left to a BCP.

Compliance: No. This should probably go to the guidelines document for SMIng module authors.

4.3.15 Hyphens in Identifiers

Type: fix

From: NMRG

Description: There has been some confusion whether hyphens are allowed in SMIv2 identifiers: Module names are allowed to contain hyphens. Node identifiers usually are not. But for example `mib-2' is a frequently used identifier that contains a hyphen due to its SMIv1 origin, when hyphen were not disallowed. Similarly, a number of named numbers of enumeration types contain hyphens violating an SMIv2 rule.

SMIng should simply allow hyphens in all kinds of identifiers. No exceptions.

Motivation: Reduce confusion and exceptions. Requires, however, that implementation mappings properly quote hyphens where appropriate.

Notes: This nit-picking is not worth to be subject to the discussion on objectives. However, SMIng should care about the fact that compilers have to map SMIng to programming languages where a hyphen is a minus and thus not allowed in identifiers.

Compliance: Ok. The language allows to use hyphens in identifiers. This is done, because there are already identifiers that contain hyphens and we want to get rid of stupid restrictions and legalized exceptions to restrictions. Mapping to programming

languages can simply be done by converting hyphens to underline characters. Underline characters are not allowed in identifiers, so that the mapping is easy and non-ambiguous.

5. Security Considerations

This document defines objectives for a language with which to write and read descriptions of management information. The language itself has no security impact on the Internet.

6. Acknowledgements

Thanks to Dave Durham, whose work on the original NIM (Network Information Model) draft was used in generating this document.

Thanks to Andrea Westerinen for her contributions on the original NIM requirements and SMIng objectives drafts.

Approximately 90% of this document has been copied from [9] to this document to ease reading and referencing the related objectives. Hence, all authors of [9] deserve our acknowledgements.

References

[1] Case, J., Fedor, M., Schoffstall, M. and J. Davin, "Simple Network Management Protocol (SNMP)", STD 15, RFC 1157, May 1990.

[2] McCloghrie, K., Case, J., Rose, M. and S. Waldbusser, "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1905, January 1996.

[3] Chan, K., Seligson, J., Durham, D., Gai, S., McCloghrie, K., Herzog, S., Reichmeyer, F., Yavatkar, R. and A. Smith, "COPS Usage for Policy Provisioning (COPS-PR)", RFC 3084, March 2001.

[4] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[5] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[6] McCloghrie, K., Perkins, D. and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

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

[8] Strauss, F., Schoenwaelder, J. and K. McCloghrie, "SMIng - Next Generation Structure of Management Information", draft-irtf-nmrg-sming-04.txt, November 2000.

[9] Elliott, C., Harrington, D., Jason, J., Schoenwaelder, J., Strauss, F. and W. Weiss, "SMIng Objectives", draft-ietf-sming-reqs-06.txt, October 2001.

[10] Strauss, F. and J. Schoenwaelder, "SMIng - Next Generation Structure of Management Information", draft-ietf-sming-02.txt, July 2001.

[11] Strauss, F. and J. Schoenwaelder, "SMIng Mappings to SNMP", draft-ietf-sming-snmp-02.txt, July 2001.

[12] Hegde, H. and R. Sahita, "SMIng Mappings to COPS-PR", draft-ietf-sming-cops-01.txt, July 2001.

[13] Strauss, F. and J. Schoenwaelder, "SMIng Core Modules", draft-ietf-sming-modules-02.txt, July 2001.

[14] Strauss, F. and J. Schoenwaelder, "SMIng Internet Protocol Core Modules", draft-ietf-sming-inet-modules-02.txt, July 2001.

Author's Address

Frank Strauss
TU Braunschweig
Muehlenpfordtstr. 23
38106 Braunschweig
Germany

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

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.