Network Working Group F. Strauss
Internet-Draft J. Schoenwaelder
Expires: January 18, 2002 TU Braunschweig
July 20, 2001

SMIng Core Modules
draft-ietf-sming-modules-02

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 18, 2002.

Copyright Notice

Abstract

This memo presents an SMIng module that introduces core data types such as counters, date and time related types, and various string types. These definitions build on RFC 2578 and RFC 2579.

Table of Contents

1. Introduction

SMIng [1] modules are built on top of some core definitions. These core definitions are imported from some "well-defined" core modules described in this memo.

The IETF-SMING module defines a set of common SMIng data types. These data types are generally applicable for modelling all areas of management information. Among these types are counter types, string types and date and time related types. This module is derived from RFC 2578 [3] and [4].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [2].

2. IETF-SMING

```
module IETF-SMING {

    organization    "IETF Next Generation Structure of
                     Management Information Working Group (SMING)";

    contact         "Frank Strauss

                     TU Braunschweig
                     Bueltenweg 74/75
                     38106 Braunschweig
                     Germany

                     Phone:  +49 531 391-3266
                     EMail:  strauss@ibr.cs.tu-bs.de";

    description     "Core type definitions for SMIng. Several
                     type definitions are SMIng versions of
                     similar SMIv2 or SPPI definitions.";

    revision {
        date        "2001-03-02";
        description "Initial revision, published as RFC &rfc.number;.";
    };

    typedef Gauge32 {
        type        Unsigned32;
        description
           "The Gauge32 type represents a non-negative integer,
            which may increase or decrease, but shall never
            exceed a maximum value, nor fall below a minimum
```

```
               value.  The maximum value can not be greater than
               2^32-1 (4294967295 decimal), and the minimum value
               can not be smaller than 0.  The value of a Gauge32
               has its maximum value whenever the information
               being modeled is greater than or equal to its
               maximum value, and has its minimum value whenever
               the information being modeled is smaller than or
               equal to its minimum value.  If the information
               being modeled subsequently decreases below
               (increases above) the maximum (minimum) value, the
               Gauge32 also decreases (increases).  (Note that
               despite of the use of the term `latched' in the
               original definition of this type, it does not
               become `stuck' at its maximum or minimum value.)";
           reference
              "RFC 2578, Sections 2. and 7.1.7.";
       };

       typedef Counter32 {
           type        Unsigned32;
           description
              "The Counter32 type represents a non-negative integer
               which monotonically increases until it reaches a
               maximum value of 2^32-1 (4294967295 decimal), when it
               wraps around and starts increasing again from zero.

               Counters have no defined `initial' value, and thus, a
               single value of a Counter has (in general) no
               information content.  Discontinuities in the
               monotonically increasing value normally occur at
               re-initialization of the management system, and at
               other times as specified in the description of an
               attribute using this type.  If such other times can
               occur, for example, the creation of a class
               instance that contains an attribute of type Counter32
               at times other than re-initialization, then
               a corresponding attribute should be defined, with an
               appropriate type, to indicate the last discontinuity.
               Examples of appropriate types include: TimeStamp,
               DateAndTime or TimeTicks (other types defined in this
               module).

               The value of the access statement for attributes with
               a type value of Counter32 should be either `readonly'
               or `eventonly'.

               A default statement should not be used for attributes
               with a type value of Counter32.";
```

```
           reference
              "RFC 2578, Sections 2. and 7.1.6.";
       };

       typedef Gauge64 {
           type        Unsigned64;
           description
              "The Gauge64 type represents a non-negative integer,
               which may increase or decrease, but shall never
               exceed a maximum value, nor fall below a minimum
               value.  The maximum value can not be greater than
               2^64-1 (18446744073709551615), and the minimum value
               can not be smaller than 0.  The value of a Gauge64
               has its maximum value whenever the information
               being modeled is greater than or equal to its
               maximum value, and has its minimum value whenever
               the information being modeled is smaller than or
               equal to its minimum value.  If the information
               being modeled subsequently decreases below
               (increases above) the maximum (minimum) value, the
               Gauge64 also decreases (increases).  (Note that
               despite of the use of the term `latched' in the
               original definition of this type, it does not
               become `stuck' at its maximum or minimum value.)";
       };

       typedef Counter64 {
           type        Unsigned64;
           description
              "The Counter64 type represents a non-negative integer
               which monotonically increases until it reaches a
               maximum value of 2^64-1 (18446744073709551615), when
               it wraps around and starts increasing again from zero.

               Counters have no defined `initial' value, and thus, a
               single value of a Counter has (in general) no
               information content.  Discontinuities in the
               monotonically increasing value normally occur at
               re-initialization of the management system, and at
               other times as specified in the description of an
               attribute using this type.  If such other times can
               occur, for example, the creation of a class
               instance that contains an attribute of type Counter32
               at times other than re-initialization, then
               a corresponding attribute should be defined, with an
               appropriate type, to indicate the last discontinuity.
               Examples of appropriate types include: TimeStamp,
               DateAndTime or TimeTicks (other types defined in this
```

```
               module).

               The value of the access statement for attributes with
               a type value of Counter64 should be either `readonly'
               or `eventonly'.

               A default statement should not be used for attributes
               with a type value of Counter64.";
           reference
              "RFC 2578, Sections 2. and 7.1.10.";
       };

       typedef Opaque {
           type        OctetString;
           status      obsolete;
           description
              "******* THIS TYPE DEFINITION IS OBSOLETE *******

               The Opaque type is provided solely for
               backward-compatibility, and shall not be used for
               newly-defined attributes and derived types.

               The Opaque type supports the capability to pass
               arbitrary ASN.1 syntax.  A value is encoded using
               the ASN.1 Basic Encoding Rules into a string of
               octets.  This, in turn, is encoded as an
               OctetString, in effect `double-wrapping' the
               original ASN.1 value.

               Note that a conforming implementation need only be
               able to accept and recognize opaquely-encoded data.
               It need not be able to unwrap the data and then
               interpret its contents.

               A requirement on `standard' modules is that no
               attribute may have a type value of Opaque and no
               type may be derived from the Opaque type.";
           reference
              "RFC 2578, Sections 2. and 7.1.9.";
       };

       typedef IpAddress {
           type        OctetString (4);
           status      deprecated;
           description
              "******* THIS TYPE DEFINITION IS DEPRECATED *******

               The IpAddress type represents a 32-bit internet
```

```
               IPv4 address.  It is represented as an OctetString
               of length 4, in network byte-order.

               Note that the IpAddress type is present for
               historical reasons. IPv4 and IPv6 addresses should
               be represented using the InetNetworkEndpoint class
               defined in the IETF-INET module.";
           reference
              "RFC 2578, Sections 2. and 7.1.5.";
       };

       typedef TimeTicks {
           type        Unsigned32;
           description
              "The TimeTicks type represents a non-negative
               integer which represents the time, modulo 2^32
               (4294967296 decimal), in hundredths of a second
               between two epochs.  When attributes are defined which
               use this type, the description of the attribute
               identifies both of the reference epochs.

               For example, the TimeStamp type (defined in this
               module) is based on the TimeTicks type.";
           reference
              "RFC 2578, Sections 2. and 7.1.8.";
       };

       typedef TimeStamp {
           type        TimeTicks;
           description
              "The value of the sysUpTime attribute at which a specific
               occurrence happened.  The specific occurrence must be
               defined in the description of any attribute defined using this
               type.  When the specific occurrence occurred prior to the
               last time sysUpTime was zero, then the TimeStamp value is
               zero.  Note that this requires all TimeStamp values to be
               reset to zero when the value of sysUpTime reaches 497+ days
               and wraps around to zero.";
           reference
              "RFC 2579, Section 2.";
       };

       typedef TimeInterval {
           type        Integer32 (0..2147483647);
           description
              "A period of time, measured in units of 0.01 seconds.

               The TimeInterval type uses Integer32 rather than
```

```
               Unsigned32 for compatibility with RFC 2579.";
           reference
              "RFC 2579, Section 2.";
       };

       typedef DateAndTime {
           type        OctetString (8 | 11);
           default     0x0000000000000000000000;
           format      "2d-1d-1d,1d:1d:1d.1d,1a1d:1d";
           description
              "A date-time specification.

               field  octets  contents                  range
               -----  ------  --------                  -----
                1      1-2   year*                     0..65536
                2       3    month                     1..12
                3       4    day                       1..31
                4       5    hour                      0..23
                5       6    minutes                   0..59
                6       7    seconds                   0..60
                             (use 60 for leap-second)
                7       8    deci-seconds              0..9
                8       9    direction from UTC        '+' / '-'
                9      10    hours from UTC*           0..13
               10      11    minutes from UTC          0..59

               * Notes:
               - the value of year is in big-endian encoding
               - daylight saving time in New Zealand is +13

               For example, Tuesday May 26, 1992 at 1:30:15 PM EDT would
               be displayed as:

                            1992-5-26,13:30:15.0,-4:0

               Note that if only local time is known, then timezone
               information (fields 8-10) is not present.

               The two special values of 8 or 11 zero bytes denote an
               unknown date-time specification.";
           reference
              "RFC 2579, Section 2.";
       };

       typedef TruthValue {
           type        Enumeration (true(1), false(2));
           description
              "Represents a boolean value.";
```

```
            reference
               "RFC 2579, Section 2.";
        };

        typedef PhysAddress {
            type        OctetString;
            format      "1x:";
            description
               "Represents media- or physical-level addresses.";
            reference
               "RFC 2579, Section 2.";
        };

        typedef MacAddress {
            type        OctetString (6);
            format      "1x:";
            description
               "Represents an IEEE 802 MAC address represented in the
                `canonical' order defined by IEEE 802.1a, i.e., as if it
                were transmitted least significant bit first, even though
                802.5 (in contrast to other 802.x protocols) requires MAC
                addresses to be transmitted most significant bit first.";
            reference
               "RFC 2579, Section 2.";
        };

        // The DisplayString definition below does not impose a size
        // restriction and is thus not the same as the DisplayString
        // definition in RFC 2579. The DisplayString255 definition is
        // provided for mapping purposes.

        typedef DisplayString {
            type        OctetString;
            format      "1a";
            description
               "Represents textual information taken from the NVT ASCII
                character set, as defined in pages 4, 10-11 of RFC 854.

                To summarize RFC 854, the NVT ASCII repertoire specifies:

                 - the use of character codes 0-127 (decimal)

                 - the graphics characters (32-126) are interpreted as
                   US ASCII

                 - NUL, LF, CR, BEL, BS, HT, VT and FF have the special
                   meanings specified in RFC 854
```

```
                - the other 25 codes have no standard interpretation

                - the sequence 'CR LF' means newline

                - the sequence 'CR NUL' means carriage-return

                - an 'LF' not preceded by a 'CR' means moving to the
                  same column on the next line.

                - the sequence 'CR x' for any x other than LF or NUL is
                  illegal.  (Note that this also means that a string may
                  end with either 'CR LF' or 'CR NUL', but not with CR.)
           ";
       };

       typedef DisplayString255 {
           type        DisplayString (0..255);
           description
              "A DisplayString with a maximum length of 255 characters.
               Any attribute defined using this syntax may not exceed 255
               characters in length.

               The DisplayString255 type has the same semantics as the
               DisplayString textual convention defined in RFC 2579.";
           reference
              "RFC 2579, Section 2.";
       };

       // The Utf8String and Utf8String255 definitions below facilitate
       // internationalization. The definition is consistent with the
       // definition of SnmpAdminString in RFC 2571.

       typedef Utf8String {
           type        OctetString;
           format      "65535t";      // is there a better way ?
           description
              "A human readable string represented using the ISO/IEC IS
               10646-1 character set, encoded as an octet string using
               the UTF-8 transformation format described in RFC 2279.

               Since additional code points are added by amendments to
               the 10646 standard from time to time, implementations must
               be prepared to encounter any code point from 0x00000000 to
               0x7fffffff.  Byte sequences that do not correspond to the
               valid UTF-8 encoding of a code point or are outside this
               range are prohibited.

               The use of control codes should be avoided. When it is
```

```
               necessary to represent a newline, the control code
               sequence CR LF should be used.

               The use of leading or trailing white space should be
               avoided.

               For code points not directly supported by user interface
               hardware or software, an alternative means of entry and
               display, such as hexadecimal, may be provided.

               For information encoded in 7-bit US-ASCII, the UTF-8
               encoding is identical to the US-ASCII encoding.

               UTF-8 may require multiple bytes to represent a single
               character / code point; thus the length of a Utf8String in
               octets may be different from the number of characters
               encoded.  Similarly, size constraints refer to the number
               of encoded octets, not the number of characters
               represented by an encoding.

               Note that the size of an Utf8String is measured in octets,
               not characters.";
       };

       typedef Utf8String255 {
           type        Utf8String (0..255);
           format      "255t";
           description
              "A Utf8String with a maximum length of 255 octets.  Note
               that the size of an Utf8String is measured in octets, not
               characters.";
       };



       identity null {
           description
              "An identity used to represent null pointer values.";
       };

   };
```

3. Security Considerations

This module does not define any management objects. Instead, it defines a set of SMIng types and classes which may be used by other SMIng modules to define management objects. These data definitions

have no security impact on the Internet.

4. Acknowledgments

Some definitions in this document are derived from RFC 2578 [3] and RFC 2579 [4], which were written by K. McCloghrie, D. Perkins, J. Schoenwaelder, J. Case, M. Rose, and S. Waldbusser.

References

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

[2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, BCP 14, March 1997.

[3] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Structure of Management Information Version 2 (SMIv2)", RFC 2578, STD 59, April 1999.

[4] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Textual Conventions for SMIv2", RFC 2579, STD 59, April 1999.

Authors' Addresses

Frank Strauss
TU Braunschweig
Bueltenweg 74/75
38106 Braunschweig
Germany

Phone: +49 531 391-3266
EMail: strauss@ibr.cs.tu-bs.de
URI: http://www.ibr.cs.tu-bs.de/

Juergen Schoenwaelder
TU Braunschweig
Bueltenweg 74/75
38106 Braunschweig
Germany

Phone: +49 531 391-3289
EMail: schoenw@ibr.cs.tu-bs.de
URI: http://www.ibr.cs.tu-bs.de/

Appendix A. OPEN ISSUES

What else is missing? - There might be more core type or class definitions that should go into the IETF-SMING module. Things that come to mind are types for Roles and RoleCombinations or types for Tags and TagLists.

Split Core Module? - Should the SMIng core module be split into several smaller modules each focussing on a specific aspect (e.g. strings, date and time, ...)?

TimeStamp - The description of the TimeStamp type builds on sysUpTime.

TimeInterval - Define TimeInterval based on Unsigned32 and remove the last sentence from the description?

Ugly Formats like `255t' - A better solution for e.g. the `255t' length restriction of Utf8String255?

Class Definitions - Maybe, we should define useful classes, like one which combines RowStatus and StorageType (and perhaps even an OwnerString)?

Full Copyright Statement

Acknowledgement

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