The SUM Pseudotype for SNMPv2 SMI

<draft-perkins-sum-00.txt>

May 27, 1996

David T. Perkins
dperkins@scruznet.com

1. Status of this Memo

This document is an Internet Draft. 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. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress."

To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the internet-drafts Shadow Directories on:

ftp.is.co.za (Africa)
nic.nordu.net (Europe)
ds.internic.net (US East Coast)
ftp.isi.edu (US West Coast)
munnari.oz.au (Pacific Rim)

2. Introduction

This memo is experimental. It specifies a pseudotype for the SNMPv2 SMI[1][2][3] to ease the development and improve the understanding of SNMP MIBs. This pseudotype is called "SUM" and has the look, and function of an ASN.1 type in the following constructs, allowed in SNMPv2 MIB modules: OBJECT-TYPE, SEQUENCE, MODULE-COMPLIANCE, AGENT-CAPABILITIES, TEXTUAL-CONVENTION, and type assignments. This pseudotype is meant to replace the current construct for specifying bit strings encoded as integers in SNMP MIBs. Existing MIBs can be easily converted to use this new construct, and MIBs that use the new construct can be converted to MIBs that are valid under the rules of the SMNPv2 SMI (and SNMPv1 SMI[4][5].) (An example of an object definition using a bit string encoded as an integer is "sysServices" found in RFCs 1213[6] and 1907[7].)

This memo proposes that the SUM pseudotype be integrated into a future version of the SNMPv2 SMI. A definition of the pseudotype is given using the ASN.1 macro notation. Also included in this memo is the extended BNF notation describing the syntax for this construct, and the guidelines for conversion between MIB modules in the SMIv2 format and those using this new pseudotype.

This memo does not specify a standard for the Internet community.

3. The ASN.1 Macro for the SUM Pseudotype

```
SUM MACRO ::=
BEGIN
    -- Note: ASN.1 macro notation is too limiting to specify all the
    -- rules for usage. Additional rules are specified as comments.

    TYPE NOTATION ::=
        -- in SEQUENCES, cannot name any bits
        -- (i.e. SEQUENCE { o1 SUM, o2 Integer32 })
        empty
        -- in SYNTAX clause, must specify the bits
        | "{" NamedBits "}"

    -- The following is just to specify the syntax for values,
    -- but does not "deliver" a useful value. (The identifier
    -- must be a name of one of the named bits.)
    VALUE NOTATION ::=
        "{" NamedBitVal "}"
        <VALUE INTEGER(0..2147483647) ::= 0>

    -- the names of bits in a value
    NamedBitVal ::=
          identifier
        | NamedBitVal "," identifier

    -- the names of bits in the type definition
    NamedBits ::=
          NamedBit
        | NamedBits "," NamedBit

    -- The bits are named and identified by their position in the
    -- integer. Bit zero is the low order (or right-most)
    -- bit in the integer. Bit 30 is the highest order allowed bit
    -- in the integer.  For an instance of a SUM macro, all named
    -- bits for that instance must be unique and the names must start
    -- with a lowercase letter. Also, all the positions for that
    -- instance must be unique and range from zero to the last one
    -- specified, which can be at most 30.  The bits do not need to
    -- be named in any order; however, all positions for bits must
    -- be specified.  The identifier must consist of one or more
    -- letters or digits (no hyphens), up to a maximum of 64
    -- characters, and the initial character must be a lower-case
    -- letter.
    NamedBit ::= identifier "(" number ")"
END
```

4. The extended BNF for the SUM Pseudotype

When used as a type, the SUM pseudotype is defined by production <sumTypeRef>. When used as a value, the SUM pseudotype is defined by production <sumValueRef>.

```
    <sumTypeRef> =
            "SUM"   -- in a sequence
          | "SUM" "{" <namedBit> [ "," <namedBit> ]... "}"

    <namedBit> = identifier "(" number ")"

    <sumValueRef> = "{" [ identifier ["," identifier]... ] "}"
```

Where:

identifier must consist of one or more letters or digits (no hyphens), up to a maximum of 64 characters, and the initial character must be a lower-case letter.

number is an unsigned integer less than or equal to 30.

5. Mapping of the SUM Pseudotype

The SUM pseudotype represents an enumeration of named bits encoded in an integer. (The pseudotype looks like the ASN.1 type BIT STRING, but maps to type INTEGER(0..2147483647).) Each collection of named bits is assigned non-negative, contiguous values, starting at zero. However, the bits do not need to be named in any order in the definition of the collection as long as the resulting collection names all bits contiguously. The maximum number of bits in a collection is 31 (i.e., the highest identifiable bit is 30).

Finally, the identifier (also called a label) for a named bit must consist of one or more letters or digits (no hyphens), up to a maximum of 64 characters, and the initial character must be a lower-case letter. (However, labels longer than 32 characters are not recommended.)

Values for the SUM pseudotype are encoded as values for the ASN.1 INTEGER type, with the bits (up to 31) packed into the value. Bits are numbered by their position, starting at zero, in the integer. Bit zero is the low order (or right-most) bit in the integer. Bit 30 is the highest order allowed bit in the integer. When a value is encoded, unspecified bits, if any, must be set to zero.

6. Example usage

The SUM pseudotype can be used in the follow situations in MIB modules:

In an OBJECT-TYPE construct, for example:

```
     o1 OBJECT-TYPE
          SYNTAX         SUM { blue(0), red(1), green(2) }
          MAX-ACCESS     read-create
          STATUS         current
          DESCRIPTION    "Example object"
          DEFVAL         { { blue, green } }
     ::= { a1 1 }
```

In a TEXTUAL-CONVENTION construct, for example:

```
     T1 TEXTUAL-CONVENTION
          STATUS         current
          DESCRIPTION    "Example textual convention"
          SYNTAX         SUM { fire(0), wind(1), rain(2) }
```

In a type assignment, for example:

```
     T1 ::= SUM { smooth(0), flexible(1), warm(2) }
```

In a SEQUENCE definition, for example:

```
     S1Entry ::= SEQUENCE {
          o1 SUM,
          o2 Integer32 }
```

In a MODULE-COMPLIANCE construct, for example:

```
     mc1 MODULE-COMPLIANCE
          STATUS         current
          DESCRIPTION    "Example module compliance"
          MODULE
            MANDATORY-GROUPS { g1 }
            OBJECT o1
              SYNTAX          SUM { fire(0), wind(1) }
              WRITE-SYNTAX    SUM { fire(0) }
              DESCRIPTION     "Example variation"
     ::= { mc 1 }
```

In an AGENT-CAPABILITIES construct, for example:

```
ac1 AGENT-CAPABILITIES
     PRODUCT-RELEASE     "Example product"
     STATUS              current
     DESCRIPTION         "Example agent capabilities"
     SUPPORTS       M1
       INCLUDES { g1 }
       VARIATION o1
         SYNTAX          SUM { fire(0), wind(1) }
         WRITE-SYNTAX    SUM { fire(0) }
         DEFVAL          { { wind } }
::= { ac 1 }
```

7. Conversion Between Current Usage and the SUM Pseudotype

In both SMIv1 and SMv2 MIB modules, the type notation for a bit string encoded in an integer value is written as "INTEGER(0..n)" (or "Integer32(0..n)"), where the value of "n" is 2 raised to the number of bits in the bit string, minus 1. For example, if there are 3 bits in the bit string, then the value of "n" is (2^3)-1, or 7. Conversion from SNMPv2 or SNMPv1 MIB modules first requires a careful review of the OBJECT-TYPE, SEQUENCE, MODULE-COMPLIANCE, AGENT-CAPABILITIES, TEXTUAL-CONVENTION, and type assignment constructs to determine which are using a bit string encoded as an integer. For each identified usage, all the bits in the string must be assigned a label. Finally, the text of the DESCRIPTION clause may need to be updated, typically to simplify.

The example below shows the definition of sysServices from RFC 1907, and a modified version using the SUM pseudotype.

From RFC 1907:

```
sysServices OBJECT-TYPE
    SYNTAX      INTEGER (0..127)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "A value which indicates the set of services that this
        entity may potentially offers.  The value is a sum.  This
        sum initially takes the value zero, Then, for each layer, L,
        in the range 1 through 7, that this node performs
        transactions for, 2 raised to (L - 1) is added to the sum.
        For example, a node which performs only routing functions
        would have a value of 4 (2^(3-1)).  In contrast, a node
        which is a host offering application services would have a
```

value of 72 (2^(4-1) + 2^(7-1)). Note that in the context of the Internet suite of protocols, values should be

```
            calculated accordingly:

                 layer      functionality
                   1        physical (e.g., repeaters)
                   2        datalink/subnetwork (e.g., bridges)
                   3        internet (e.g., supports the IP)
                   4        end-to-end  (e.g., supports the TCP)
                   7        applications (e.g., supports the SMTP)

            For systems including OSI protocols, layers 5 and 6 may also
            be counted."
        ::= { system 7 }
```

Modified version using the SUM pseudotype:

```
    sysServices OBJECT-TYPE
        SYNTAX      SUM { physical(0), datalinkOrSubnetwork(1),
                          internet(2), endToEnd(3), session(4),
                          presentation(5), applications(6) }
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The set of services that this entity may potentially
            offer.  The members of this set are the protocol 'layers'
            for which this node performs transactions.  For example,
            a node which performs only routing functions would have
            a value of { internet }.  In contrast, a node which is
            a host offering application services would have a value
            of { endToEnd, applications }.  Note that in the
            context of the Internet suite of protocols, the
            bits should be interpreted accordingly:

                 layer      functionality
                   1        physical(0) (e.g., repeaters)
                   2        datalinkOrSubnetwork(1) (e.g., bridges)
                   3        internet(2) (e.g., supports the IP)
                   4        endToEnd(3)  (e.g., supports the TCP)
                   7        applications(6) (e.g., supports the SMTP)

            For systems including OSI protocols, layers 5 and 6
            should be interpreted normally."
        ::= { system 7 }
```

Conversion of the SUM pseudotype back to an INTEGER (or Integer32) is quite trivial.

8. Allowed Updates

An instance of an SUM pseudotype may not be changed. A modification requires that a new instance of the construct in which it is used be created. (Note that requirement is consistent with that specified in section 10.2 of SMIv2.)

9. Acknowledgments

Thanks go to Bancroft Scott for his assistance on the BITS macro on which the SUM macro is modeled.

10. References

[1] J. Case, K. McCloghrie, M. Rose, S. Waldbusser, "Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1902, 01/22/1996.

[2] J. Case, K. McCloghrie, M. Rose, S. Waldbusser, "Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1903, 01/22/1996.

[3] J. Case, K. McCloghrie, M. Rose, S. Waldbusser, "Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1904, 01/22/1996.

[4] K. McCloghrie, M. Rose, "Structure and Identification of Management Information for TCP/IP-based Internets", RFC 1155, 05/10/1990.

[5] K. McCloghrie, M. Rose, "Concise MIB Definitions", RFC 1212, 03/26/1991.

[6] K. McCloghrie, M. Rose, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", 03/26/1991.

[7] J. Case, K. McCloghrie, M. Rose, S. Waldbusser, "Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)", 01/22/1996