The ENUM PseudoType for SNMPv2 SMI

<draft-perkins-enum-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 "ENUM" 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 enumerated 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].)

This memo proposes that the ENUM 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 ENUM Pseudotype

```
ENUM 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 values
        -- (i.e. SEQUENCE { o1 ENUM, o2 Integer32 })
        empty
        -- in SYNTAX clause, must specify the named (enumerated) values
        | "{" EnumValues "}"

    -- 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 an enumerated values.)
    VALUE NOTATION ::=
        identifier
        <VALUE INTEGER(-2147483648..2147483647) ::= 0>

    -- the enumerated values (named values) in the type definition
    EnumValues ::=
          EnumValue
        | EnumValues "," EnumValue

    -- The enumerated (named) values can be any in the range
    -- of -2147483648 to 2147483647.  Each name must be unique
    -- for the type definition.  Only those named values may be
    -- members of the range.  Note that though it is recommended that
    -- enumerated values start at 1 and be numbered contiguously,
    -- any valid value for Integer32 is allowed for an enumerated
    -- value and, further, enumerated values needn't be contiguously
    -- assigned.
    EnumValue ::= identifier "(" IntVal")"

    IntVal ::= "-" number | number
END
```

4. The extended BNF for the ENUM Pseudotype

When used as a type, the ENUM pseudotype is defined by production <EnumTypeRef>. When used as a value, the ENUM pseudotype is defined by production <EnumValueRef>.

```
    <EnumTypeRef> =
            "ENUM"   -- in a sequence
          | "ENUM" "{" <namedValue> [ "," <namedValue> ]... "}"

    <namedValue> = identifier "(" <intValue> ")"

    <intValue> = "-" number | number

    <EnumValueRef> = 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 2^31 (2147483648) when preceded by a minus sign, otherwise number must be less than 2^31 (2147483648).

5. Mapping of the ENUM Pseudotype

The ENUM pseudotype represents an enumeration of named values. (The pseudotype looks like the ASN.1 type ENUMERATED, but maps to type INTEGER(-2147483648..2147483647).) Only those named values from the range of integer values may be present as a value. Note that although it is recommended that enumerated values start at 1 and be numbered contiguously, any valid value for Integer32 (a pseudonym for type INTEGER(-2147483648..2147483647)) is allowed for an enumerated value and, further, enumerated values needn't be contiguously assigned.

Finally, the identifier (also called a label) for an enumeration 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 ENUM pseudotype are encoded as values for the ASN.1 INTEGER type.

6. Example Usage

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

In an OBJECT-TYPE construct, for example:

```
o1 OBJECT-TYPE
     SYNTAX         ENUM { monday(1), tuesday(2), wednesday(3) }
     MAX-ACCESS     read-create
     STATUS         current
     DESCRIPTION    "Example object"
     DEFVAL         { wednesday }
::= { a1 1 }
```

In a TEXTUAL-CONVENTION construct, for example:

```
T1 TEXTUAL-CONVENTION
     STATUS         current
     DESCRIPTION    "Example textual convention"
     SYNTAX         ENUM { north(10), east(20),
                           south(-10), west(-20) }
```

In a type assignment, for example:

```
T1 ::= ENUM { cool(0), warm(10), hot(17) }
```

In a SEQUENCE definition, for example:

```
S1Entry ::= SEQUENCE {
     o1 ENUM,
     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          ENUM { monday(1), tuesday(2) }
         WRITE-SYNTAX    ENUM { tuesday(2) }
         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          ENUM { tuesday(2), wednesday(3) }
            WRITE-SYNTAX    ENUM { wednesday(3) }
            DEFVAL          { wednesday }
   ::= { ac 1 }
```

7. Conversion Between Current Usage and the ENUM Pseudotype

In both SMIv1 and SMv2 MIB modules, the type notation for an enumerated integer is written as "INTEGER { <namedValues> }". Conversion to and from the ENUM pseudotype consists of using the word "INTEGER" or "ENUM" in the appropriate places. This requires more that just a simple "search and replace" algorithm when converting from "INTEGER" to "ENUM" due to ambiguity in the SEQUENCE construct. (Note that this is not a claim that the SMIv1 and SMv2 MIB module language is ambiguous. It is only to point out that the syntax of objects specified in a SEQUENCE construct must be known.) The algorithm converting "ENUM" to "INTEGER" is a simple replacement of "ENUM" tokens with "INTEGER" tokens. Both conversion algorithms need to also adjust the IMPORTS clause, since use of the "INTEGER" construct requires no imports and the "ENUM" requires an import from the SMI.

8. Allowed Updates

An instance of an ENUM pseudotype may have new enumerations added or existing labels changed when a MIB module is updated. (Note that this allowance 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 ENUM 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.