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 "iid-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 informational. It specifies an approach to add 64-bit signed and unsigned integer types to versions 1 and 2 of the SNMP SMI[1][2][3][4][5][6], and versions 1 and 2 of the SNMP protocol[7][8][9] without changes. Thus, this addition requires no modifications to existing SNMP MIB compilers, and no changes to existing SNMP protocol engines used in SNMP agents and SNMP management applications.

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

3. Background

The SNMP protocol and SMI is based on elements from ASN.1[10] and BER[11]. The SMI allows use of only a few ASN.1 base types plus a few SNMP application specific types. Support for large integer types is not currently found in SNMP. This is primarily due to two factors. The first was the focus during the original development of SNMP to keep SNMP simple and use it for managing computer network devices using the Internet protocol suite. The second factor was to limit the resource requirements for agents to support SNMP.

SNMP has been found to be useful for purposes other than those for which it was originally developed. However, some of the limitations in SNMP have restricted its continuing growth. Lack of support for large integer types has been a problem in some areas. The first example is in mid-level managers that gather and process management information from many sources. The processing includes computing values for mathematical formulas that require use of 64-bit integer arithmetic. Mid-level managers typically execute on general purpose computers. Some have direct support for 64-bit integers, and all should have access to libraries to support multi-precision arithmetic. Thus, supporting 64-bit integer types is not a burden for them. A second area is using SNMP in equipment that require 64-bit integer precision objects to monitor, control, and configure them due to very high-speed network interfaces, such as ATM, and giga-bit ethernet.

4. Large Integer Types

ASN.1 already has support for large integer data types. However, the SNMP SMI restricts their range. Thus, they cannot be used directly. The current need for large integer data types is satisfied by the addition of a 64-bit signed integer, and a 64-bit unsigned integer data type, which are called Integer64 and Unsigned64, respectively, in this memo. Section 5.1.2 of "The Domestication of the Opaque Type for SNMP"[12] requires that a new base type be identified and a textual
convention be defined for each new "wrapped" type. Shown below are the
definitions for these types and corresponding textual conventions.

-- A signed integer of up to 64 bits of precision.
I64Type ::= [APPLICATION 10] IMPLICIT INTEGER (-9223372036854775808..9223372036854775807)

-- An unsigned integer of up to 64 bit of precision.
U64Type ::= [APPLICATION 11] IMPLICIT INTEGER (0..18446744073709551615)

Integer64 TEXTUAL-CONVENTION
STATUS current
DESCRIPTION
"A 64-bit signed integer. The value is restricted to
the BER serialization of the following ASN.1 type:
I64TYPE ::= [122] IMPLICIT I64Type
(note: the value 122 is the sum of ‘30’h and ‘4a’h)
The BER serialization of the length for values of
this type must use the definite length, short
encoding form.

For example, the BER serialization of value 129
of type I64TYPE is ‘9f7a020081’h. (The tag is ‘9f7a’h;
the length is ‘02’h; and the value is ‘0081’h.) The
BER serialization of value ‘9f7a020081’h of data
type Opaque is ‘44059f7a020081’h. (The tag is ‘44’h;
the length is ‘05’h; and the value is ‘9f7a020081’h.)
Also for example, the BER serialization of value -129
of type I64TYPE is ‘9f7a02ff7f’h. (The tag is ‘9f7a’h;
the length is ‘02’h; and the value is ‘ff7f’h.) The
BER serialization of value ‘9f7a02ff7f’h of data
type Opaque is ‘44059f7a02ff7f’h. (The tag is ‘44’h;
the length is ‘05’h; and the value is ‘9f7a02ff7f’h.)"
SYNTAX Opaque (SIZE(4..11))

Unsigned64 TEXTUAL-CONVENTION
STATUS current
DESCRIPTION
"A 64-bit unsigned signed integer. The value is
restricted to the BER serialization of the following
ASN.1 type:
U64TYPE ::= [123] IMPLICIT U64Type
(note: the value 123 is the sum of ‘30’h and ‘4b’h)
The BER serialization of the length for values of
this type must use the definite length, short
encoding form.
For example, the BER serialization of value 129 of type U64TYPE is '9f7b020081'h. (The tag is '9f7b'h; the length is '02'h; and the value is '0081'h.) The BER serialization of value '9f7b020081'h of data type Opaque is '44059f7b020081'h. (The tag is '44'h; the length is '05'h; and the value is '9f7b020081'h.)

SYNTAX     Opaque (SIZE(4..12))

6. References


