**Protocol Operations
for Version 2 of the
Simple Network Management Protocol (SNMPv2)**

11 October 1995 |

draft-ietf-snmpv2-proto-ds-05.txt |

Keith McCloghrie
Editor   +
Cisco Systems, Inc.
kzm@cisco.com

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
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.''

To learn the current status of any Internet-Draft, please check the
``1id-abstracts.txt'' listing contained in the Internet- Drafts Shadow
Directories on ds.internic.net (US East Coast), nic.nordu.net (Europe),
ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

1. Introduction

A management system contains: several (potentially many) nodes, each
with a processing entity, termed an agent, which has access to
management instrumentation; at least one management station; and, a
management protocol, used to convey management information between the
agents and management stations. Operations of the protocol are carried
out under an administrative framework which defines authentication,
authorization, access control, and privacy policies.

Management stations execute management applications which monitor and
control managed elements. Managed elements are devices such as hosts,
routers, terminal servers, etc., which are monitored and controlled via
access to their management information.

Management information is viewed as a collection of managed objects,
residing in a virtual information store, termed the Management
Information Base (MIB). Collections of related objects are defined in
MIB modules. These modules are written using a subset of OSI's Abstract
Syntax Notation One (ASN.1) [1], termed the Structure of Management
Information (SMI) [2].

The management protocol, version 2 of the Simple Network Management
Protocol, provides for the exchange of messages which convey management
information between the agents and the management stations. The form of
these messages is a message "wrapper" which encapsulates a Protocol Data
Unit (PDU). The form and meaning of the "wrapper" is determined by an
administrative framework which defines both authentication and
authorization policies.

It is the purpose of this document, Protocol Operations for SNMPv2, to
define the operations of the protocol with respect to the sending and
receiving of the PDUs.

1.1. A Note on Terminology

For the purpose of exposition, the original Internet-standard Network
Management Framework, as described in RFCs 1155, 1157, and 1212, is
termed the SNMP version 1 framework (SNMPv1). The current framework is |
termed the SNMP version 2 framework (SNMPv2), as described in [3]. |

2. **Overview**

2.1. **Roles of Protocol Entities**

A SNMPv2 entity may operate in a manager role or an agent role.

A SNMPv2 entity acts in an agent role when it performs SNMPv2 management
operations in response to received SNMPv2 protocol messages (other than
an inform notification) or when it sends trap notifications.

A SNMPv2 entity acts in a manager role when it initiates SNMPv2
management operations by the generation of SNMPv2 protocol messages or
when it performs SNMPv2 management operations in response to received
trap or inform notifications.

A SNMPv2 entity may support either or both roles, as dictated by its
implementation and configuration. Further, a SNMPv2 entity can also act
in the role of a proxy agent, in which it appears to be acting in an
agent role, but satisfies management requests by acting in a manager
role with a remote entity.

2.2. **Management Information**

The term, variable, refers to an instance of a non-aggregate object type
defined according to the conventions set forth in the SMI [2] or the
textual conventions based on the SMI [4]. The term, variable binding,
normally refers to the pairing of the name of a variable and its
associated value. However, if certain kinds of exceptional conditions
occur during processing of a retrieval request, a variable binding will
pair a name and an indication of that exception.

A variable-binding list is a simple list of variable bindings.

The name of a variable is an OBJECT IDENTIFIER which is the
concatenation of the OBJECT IDENTIFIER of the corresponding object-type
together with an OBJECT IDENTIFIER fragment identifying the instance.
The OBJECT IDENTIFIER of the corresponding object-type is called the
OBJECT IDENTIFIER prefix of the variable.

2.3. **Access to Management Information**

Three types of access to management information are provided by the
protocol. One type is a request-response interaction, in which a SNMPv2

entity, acting in a manager role, sends a request to a SNMPv2 entity,
acting in an agent role, and the latter SNMPv2 entity then responds to
the request. This type is used to retrieve or modify management
information associated with the managed device.

A second type is also a request-response interaction, in which a SNMPv2
entity, acting in a manager role, sends a request to a SNMPv2 entity,
also acting in a manager role, and the latter SNMPv2 entity then
responds to the request. This type is used to notify a SNMPv2 entity,
acting in a manager role, of management information associated with
another SNMPv2 entity, also acting in a manager role.

The third type of access is an unconfirmed interaction, in which a
SNMPv2 entity, acting in an agent role, sends a unsolicited message,
termed a trap, to a SNMPv2 entity, acting in a manager role, and no
response is returned. This type is used to notify a SNMPv2 entity,
acting in a manager role, of an exceptional situation, which has
resulted in changes to management information associated with the
managed device.

2.4. Retransmission of Requests

For all types of request in this protocol, the receiver is required
under normal circumstances, to generate and transmit a response to the
originator of the request. Whether or not a request should be
retransmitted if no corresponding response is received in an appropriate
time interval, is at the discretion of the application originating the
request. This will normally depend on the urgency of the request.
However, such an application needs to act responsibly in respect to the
frequency and duration of re-transmissions.

2.5. Message Sizes

The maximum size of a SNMPv2 message is limited to the minimum of:

(1) the maximum message size which the destination SNMPv2 entity can
     accept; and,

(2) the maximum message size which the source SNMPv2 entity can
     generate.

The former may be known on a per-recipient basis; and in the absence of |
such knowledge, |

is indicated by transport domain used when sending the message. The
latter is imposed by implementation-specific local constraints.

Each transport mapping for the SNMPv2 indicates the minimum message size
which a SNMPv2 implementation must be able to produce or consume.
Although implementations are encouraged to support larger values
whenever possible, a conformant implementation must never generate
messages larger than allowed by the receiving SNMPv2 entity.

One of the aims of the GetBulkRequest-PDU, specified in this protocol,
is to minimize the number of protocol exchanges required to retrieve a
large amount of management information. As such, this PDU type allows a
SNMPv2 entity acting in a manager role to request that the response be
as large as possible given the constraints on message sizes. These
constraints include the limits on the size of messages which the SNMPv2
entity acting in an agent role can generate, and the SNMPv2 entity
acting in a manager role can receive.

However, it is possible that such maximum sized messages may be larger
than the Path MTU of the path across the network traversed by the
messages. In this situation, such messages are subject to
fragmentation. Fragmentation is generally considered to be harmful [5],
since among other problems, it leads to a decrease in the reliability of
the transfer of the messages. Thus, a SNMPv2 entity which sends a
GetBulkRequest-PDU must take care to set its parameters accordingly, so
as to reduce the risk of fragmentation. In particular, under conditions
of network stress, only small values should be used for max-repetitions.

2.6. **Transport Mappings**

It is important to note that the exchange of SNMPv2 messages requires
only an unreliable datagram service, with every message being entirely
and independently contained in a single transport datagram. Specific
transport mappings and encoding rules are specified elsewhere [6].
However, the preferred mapping is the use of the User Datagram Protocol
[7].

[3](#). **Definitions**

```
SNMPv2-PDU DEFINITIONS ::= BEGIN

IMPORTS
    ObjectName, ObjectSyntax, Integer32
        FROM SNMPv2-SMI;


-- protocol data units

PDUs ::=
    CHOICE {
        get-request
            GetRequest-PDU,

        get-next-request
            GetNextRequest-PDU,

        get-bulk-request
            GetBulkRequest-PDU,

        response
            Response-PDU,

        set-request
            SetRequest-PDU,

        inform-request
            InformRequest-PDU,

        snmpV2-trap
            SNMPv2-Trap-PDU,

        report
            Report-PDU,
    }
```

```
-- PDUs

GetRequest-PDU ::=
    [0]
        IMPLICIT PDU

GetNextRequest-PDU ::=
    [1]
        IMPLICIT PDU

Response-PDU ::=
    [2]
        IMPLICIT PDU

SetRequest-PDU ::=
    [3]
        IMPLICIT PDU

-- [4] is obsolete

GetBulkRequest-PDU ::=
    [5]
        IMPLICIT BulkPDU

InformRequest-PDU ::=
    [6]
        IMPLICIT PDU

SNMPv2-Trap-PDU ::=
    [7]
        IMPLICIT PDU

--   Usage and precise semantics of Report-PDU are not presently    |
--   defined.  Any SNMP administrative framework making use of      |
--   this PDU must define its usage and semantics.                  |
Report-PDU ::=
    [8]
        IMPLICIT PDU
```

```
max-bindings
    INTEGER ::= 2147483647

PDU ::=
    SEQUENCE {
        request-id
            Integer32,

        error-status            -- sometimes ignored
            INTEGER {
                noError(0),
                tooBig(1),
                noSuchName(2),   -- for proxy compatibility
                badValue(3),     -- for proxy compatibility
                readOnly(4),     -- for proxy compatibility
                genErr(5),
                noAccess(6),
                wrongType(7),
                wrongLength(8),
                wrongEncoding(9),
                wrongValue(10),
                noCreation(11),
                inconsistentValue(12),
                resourceUnavailable(13),
                commitFailed(14),
                undoFailed(15),
                authorizationError(16),
                notWritable(17),
                inconsistentName(18)
            },

        error-index             -- sometimes ignored
            INTEGER (0..max-bindings),

        variable-bindings   -- values are sometimes ignored
            VarBindList
    }
```

```
    BulkPDU ::=                          -- MUST be identical in
        SEQUENCE {                       -- structure to PDU
            request-id
                Integer32,

            non-repeaters
                INTEGER (0..max-bindings),

            max-repetitions
                INTEGER (0..max-bindings),

            variable-bindings      -- values are ignored
                VarBindList
        }
```

```
    -- variable binding

    VarBind ::=
        SEQUENCE {
            name
                ObjectName,

            CHOICE {
                value
                    ObjectSyntax,

                unSpecified        -- in retrieval requests
                        NULL,

                                   -- exceptions in responses
                noSuchObject[0]
                        IMPLICIT NULL,

                noSuchInstance[1]
                        IMPLICIT NULL,

                endOfMibView[2]
                        IMPLICIT NULL
            }
        }


    -- variable-binding list

    VarBindList ::=
        SEQUENCE (SIZE (0..max-bindings)) OF
            VarBind


    END
```

4. Protocol Specification

4.1. Common Constructs

The value of the request-id field in a Response-PDU takes the value of
the request-id field in the request PDU to which it is a response. By
use of the request-id value, a SNMPv2 application can distinguish the
(potentially multiple) outstanding requests, and thereby correlate
incoming responses with outstanding requests. In cases where an
unreliable datagram service is used, the request-id also provides a
simple means of identifying messages duplicated by the network. Use of
the same request-id on a retransmission of a request allows the response
to either the original transmission or the retransmission to satisfy the
request. However, in order to calculate the round trip time for
transmission and processing of a request-response transaction, the
SNMPv2 application needs to use a different request-id value on a
retransmitted request. The latter strategy is recommended for use in
the majority of situations.

A non-zero value of the error-status field in a Response-PDU is used to
indicate that an exception occurred to prevent the processing of the
request. In these cases, a non-zero value of the Response-PDU's error-
index field provides additional information by identifying which
variable binding in the list caused the exception. A variable binding
is identified by its index value. The first variable binding in a
variable-binding list is index one, the second is index two, etc.

SNMPv2 limits OBJECT IDENTIFIER values to a maximum of 128 sub-
identifiers, where each sub-identifier has a maximum value of 2**32-1.

4.2. PDU Processing

It is mandatory that all SNMPv2 entities acting in an agent role be able
to generate the following PDU types: Response-PDU and SNMPv2-Trap-PDU;
further, all such implementations must be able to receive the following
PDU types: GetRequest-PDU, GetNextRequest-PDU, GetBulkRequest-PDU, and
SetRequest-PDU.

It is mandatory that all SNMPv2 entities acting in a manager role be
able to generate the following PDU types: GetRequest-PDU,
GetNextRequest-PDU, GetBulkRequest-PDU, SetRequest-PDU, InformRequest-
PDU, and Response-PDU; further, all such implementations must be able to
receive the following PDU types: Response-PDU, SNMPv2-Trap-PDU,

InformRequest-PDU;

In the elements of procedure below, any field of a PDU which is not
referenced by the relevant procedure is ignored by the receiving SNMPv2
entity. However, all components of a PDU, including those whose values
are ignored by the receiving SNMPv2 entity, must have valid ASN.1 syntax
and encoding. For example, some PDUs (e.g., the GetRequest-PDU) are
concerned only with the name of a variable and not its value. In this
case, the value portion of the variable binding is ignored by the
receiving SNMPv2 entity. The unSpecified value is defined for use as
the value portion of such bindings.

On generating a management communication, the message "wrapper" to
encapsulate the PDU is generated according to the "Elements of
Procedure" of the administrative framework in use is followed (for |
example, see [3]). |
While the definition of "max-bindings" does impose an upper-bound on the
number of variable bindings, in practice, the size of a message is
limited only by constraints on the maximum message size -- it is not
limited by the number of variable bindings.

On receiving a management communication, the "Elements of Procedure" of |
the administrative framework in use is followed (for example, see [3]), |
and if those procedures indicate that the operation contained within the
message is to be performed locally, then those procedures also indicate
the MIB view which is visible to the operation.

4.2.1. The GetRequest-PDU

A GetRequest-PDU is generated and transmitted at the request of a SNMPv2
application.

Upon receipt of a GetRequest-PDU, the receiving SNMPv2 entity processes
each variable binding in the variable-binding list to produce a
Response-PDU. All fields of the Response-PDU have the same values as
the corresponding fields of the received request except as indicated
below. Each variable binding is processed as follows:

(1) If the variable binding's name exactly matches the name of a
     variable accessible by this request, then the variable binding's
     value field is set to the value of the named variable.

(2) Otherwise, if the variable binding's name does not have an OBJECT
     IDENTIFIER prefix which exactly matches the OBJECT IDENTIFIER

prefix of any (potential) variable accessible by this request, then
its value field is set to `noSuchObject'.

(3) Otherwise, the variable binding's value field is set to to
`noSuchInstance'.

If the processing of any variable binding fails for a reason other than
listed above, then the Response-PDU is re-formatted with the same values
in its request-id and variable-bindings fields as the received
GetRequest-PDU, with the value of its error-status field set to
`genErr', and the value of its error-index field is set to the index of
the failed variable binding.

Otherwise, the value of the Response-PDU's error-status field is set to
`noError', and the value of its error-index field is zero.

The generated Response-PDU is then encapsulated into a message. If the
size of the resultant message is less than or equal to both a local
constraint and the maximum message size of the originator, it is
transmitted to the originator of the GetRequest-PDU.

Otherwise, an alternate Response-PDU is generated. This alternate
Response-PDU is formatted with the same value in its request-id field as
the received GetRequest-PDU, with the value of its error-status field
set to `tooBig', the value of its error-index field set to zero, and an
empty variable-bindings field. This alternate Response-PDU is then
encapsulated into a message. If the size of the resultant message is
less than or equal to both a local constraint and the maximum message
size of the originator, it is transmitted to the originator of the
GetRequest-PDU. Otherwise, the snmpSilentDrops |
[10] counter is incremented and the resultant message is discarded.

4.2.2. The GetNextRequest-PDU

A GetNextRequest-PDU is generated and transmitted at the request of a
SNMPv2 application.

Upon receipt of a GetNextRequest-PDU, the receiving SNMPv2 entity
processes each variable binding in the variable-binding list to produce
a Response-PDU. All fields of the Response-PDU have the same values as
the corresponding fields of the received request except as indicated
below. Each variable binding is processed as follows:

(1)   The variable is located which is in the lexicographically ordered
      list of the names of all variables which are accessible by this
      request and whose name is the first lexicographic successor of the
      variable binding's name in the incoming GetNextRequest-PDU. The
      corresponding variable binding's name and value fields in the
      Response-PDU are set to the name and value of the located variable.

(2)   If the requested variable binding's name does not lexicographically
      precede the name of any variable accessible by this request, i.e.,
      there is no lexicographic successor, then the corresponding
      variable binding produced in the Response-PDU has its value field
      set to `endOfMibView', and its name field set to the variable
      binding's name in the request.

If the processing of any variable binding fails for a reason other than
listed above, then the Response-PDU is re-formatted with the same values
in its request-id and variable-bindings fields as the received
GetNextRequest-PDU, with the value of its error-status field set to
`genErr', and the value of its error-index field is set to the index of
the failed variable binding.

Otherwise, the value of the Response-PDU's error-status field is set to
`noError', and the value of its error-index field is zero.

The generated Response-PDU is then encapsulated into a message. If the
size of the resultant message is less than or equal to both a local
constraint and the maximum message size of the originator, it is
transmitted to the originator of the GetNextRequest-PDU.

Otherwise, an alternate Response-PDU is generated. This alternate
Response-PDU is formatted with the same values in its request-id field
as the received GetNextRequest-PDU, with the value of its error-status
field set to `tooBig', the value of its error-index field set to zero,
and an empty variable-bindings field. This alternate Response-PDU is
then encapsulated into a message. If the size of the resultant message
is less than or equal to both a local constraint and the maximum message
size of the originator, it is transmitted to the originator of the
GetNextRequest-PDU. Otherwise, the snmpSilentDrops |
[10] counter is incremented and the resultant message is discarded.

4.2.2.1. Example of Table Traversal

An important use of the GetNextRequest-PDU is the traversal of
conceptual tables of information within a MIB. The semantics of this

type of request, together with the method of identifying individual
instances of objects in the MIB, provides access to related objects in
the MIB as if they enjoyed a tabular organization.

In the protocol exchange sketched below, a SNMPv2 application retrieves
the media-dependent physical address and the address-mapping type for
each entry in the IP net-to-media Address Translation Table [8] of a
particular network element. It also retrieves the value of sysUpTime
[10], at which the mappings existed. Suppose that the agent's IP net-
to-media table has three entries:

```
  Interface-Number  Network-Address  Physical-Address  Type

        1              10.0.0.51      00:00:10:01:23:45  static
        1               9.2.3.4       00:00:10:54:32:10  dynamic
        2              10.0.0.15      00:00:10:98:76:54  dynamic
```

The SNMPv2 entity acting in a manager role begins by sending a
GetNextRequest-PDU containing the indicated OBJECT IDENTIFIER values as
the requested variable names:

```
    GetNextRequest ( sysUpTime,
                     ipNetToMediaPhysAddress,
                     ipNetToMediaType )
```

The SNMPv2 entity acting in an agent role responds with a Response-PDU:

```
    Response (( sysUpTime.0 =  "123456" ),
              ( ipNetToMediaPhysAddress.1.9.2.3.4 =
                                     "000010543210" ),
              ( ipNetToMediaType.1.9.2.3.4 =  "dynamic" ))
```

The SNMPv2 entity acting in a manager role continues with:

```
    GetNextRequest ( sysUpTime,
                     ipNetToMediaPhysAddress.1.9.2.3.4,
                     ipNetToMediaType.1.9.2.3.4 )
```

The SNMPv2 entity acting in an agent role responds with:

```
    Response (( sysUpTime.0 =  "123461" ),
              ( ipNetToMediaPhysAddress.1.10.0.0.51 =
                                     "000010012345" ),
              ( ipNetToMediaType.1.10.0.0.51 =  "static" ))
```

The SNMPv2 entity acting in a manager role continues with:

    GetNextRequest (sysUpTime,
                     ipNetToMediaPhysAddress.1.10.0.0.51,
                     ipNetToMediaType.1.10.0.0.51)

The SNMPv2 entity acting in an agent role responds with:

    Response ((sysUpTime.0 = "123466"),
              (ipNetToMediaPhysAddress.2.10.0.0.15 =
                                     "000010987654"),
              (ipNetToMediaType.2.10.0.0.15 = "dynamic"))

The SNMPv2 entity acting in a manager role continues with:

    GetNextRequest (sysUpTime,
                     ipNetToMediaPhysAddress.2.10.0.0.15,
                     ipNetToMediaType.2.10.0.0.15)

As there are no further entries in the table, the SNMPv2 entity acting
in an agent role responds with the variables that are next in the
lexicographical ordering of the accessible object names, for example:

    Response ((sysUpTime.0 = "123471"),
              (ipNetToMediaNetAddress.1.9.2.3.4 =
                                     "9.2.3.4"),
              (ipRoutingDiscards.0 = "2"))

This response signals the end of the table to the SNMPv2 entity acting
in a manager role.

4.2.3. The GetBulkRequest-PDU

A GetBulkRequest-PDU is generated and transmitted at the request of a
SNMPv2 application. The purpose of the GetBulkRequest-PDU is to request
the transfer of a potentially large amount of data, including, but not
limited to, the efficient and rapid retrieval of large tables.

Upon receipt of a GetBulkRequest-PDU, the receiving SNMPv2 entity
processes each variable binding in the variable-binding list to produce
a Response-PDU with its request-id field having the same value as in the
request. Processing begins by examining the values in the non-repeaters
and max-repetitions fields. If the value in the non-repeaters field is
less than zero, then the value of the field is set to zero. Similarly,

if the value in the max-repetitions field is less than zero, then the
value of the field is set to zero.

For the GetBulkRequest-PDU type, the successful processing of each
variable binding in the request generates zero or more variable bindings
in the Response-PDU. That is, the one-to-one mapping between the
variable bindings of the GetRequest-PDU, GetNextRequest-PDU, and
SetRequest-PDU types and the resultant Response-PDUs does not apply for
the mapping between the variable bindings of a GetBulkRequest-PDU and
the resultant Response-PDU.

The values of the non-repeaters and max-repetitions fields in the
request specify the processing requested. One variable binding in the
Response-PDU is requested for the first N variable bindings in the
request and M variable bindings are requested for each of the R
remaining variable bindings in the request. Consequently, the total
number of requested variable bindings communicated by the request is
given by N + (M * R), where N is the minimum of: a) the value of the
non-repeaters field in the request, and b) the number of variable
bindings in the request; M is the value of the max-repetitions field in
the request; and R is the maximum of: a) number of variable bindings in
the request - N, and b) zero.

The receiving SNMPv2 entity produces a Response-PDU with up to the total
number of requested variable bindings communicated by the request. The
request-id shall have the same value as the received GetBulkRequest-PDU.

If N is greater than zero, the first through the (N)-th variable
bindings of the Response-PDU are each produced as follows:

(1) The variable is located which is in the lexicographically ordered
     list of the names of all variables which are accessible by this
     request and whose name is the first lexicographic successor of the
     variable binding's name in the incoming GetBulkRequest-PDU. The
     corresponding variable binding's name and value fields in the
     Response-PDU are set to the name and value of the located variable.

(2) If the requested variable binding's name does not lexicographically
     precede the name of any variable accessible by this request, i.e.,
     there is no lexicographic successor, then the corresponding
     variable binding produced in the Response-PDU has its value field
     set to `endOfMibView', and its name field set to the variable
     binding's name in the request.

If M and R are non-zero, the (N + 1)-th and subsequent variable bindings
of the Response-PDU are each produced in a similar manner. For each
iteration i, such that i is greater than zero and less than or equal to
M, and for each repeated variable, r, such that r is greater than zero
and less than or equal to R, the (N + ((i-1) * R) + r)-th variable
binding of the Response-PDU is produced as follows:

(1) The variable which is in the lexicographically ordered list of the
     names of all variables which are accessible by this request and
     whose name is the (i)-th lexicographic successor of the (N + r)-th
     variable binding's name in the incoming GetBulkRequest-PDU is
     located and the variable binding's name and value fields are set to
     the name and value of the located variable.

(2) If there is no (i)-th lexicographic successor, then the
     corresponding variable binding produced in the Response-PDU has its
     value field set to `endOfMibView', and its name field set to either
     the last lexicographic successor, or if there are no lexicographic
     successors, to the (N + r)-th variable binding's name in the
     request.

While the maximum number of variable bindings in the Response-PDU is
bounded by N + (M * R), the response may be generated with a lesser
number of variable bindings (possibly zero) for either of three reasons.

(1) If the size of the message encapsulating the Response-PDU
     containing the requested number of variable bindings would be
     greater than either a local constraint or the maximum message size
     of the originator, then the response is generated with a lesser
     number of variable bindings. This lesser number is the ordered set
     of variable bindings with some of the variable bindings at the end
     of the set removed, such that the size of the message encapsulating
     the Response-PDU is approximately equal to but no greater than
     either a local constraint or the maximum message size of the
     originator. Note that the number of variable bindings removed has
     no relationship to the values of N, M, or R.

(2) The response may also be generated with a lesser number of variable
     bindings if for some value of iteration i, such that i is greater
     than zero and less than or equal to M, that all of the generated
     variable bindings have the value field set to the `endOfMibView'.
     In this case, the variable bindings may be truncated after the (N +
     (i * R))-th variable binding.

(3)   In the event that the processing of a request with many repetitions
      requires a significantly greater amount of processing time than a
      normal request, then an agent may terminate the request with less
      than the full number of repetitions, providing at least one
      repetition is completed.

If the processing of any variable binding fails for a reason other than
listed above, then the Response-PDU is re-formatted with the same values
in its request-id and variable-bindings fields as the received
GetBulkRequest-PDU, with the value of its error-status field set to
`genErr', and the value of its error-index field is set to the index of
the variable binding in the original request which corresponds to the
failed variable binding.

Otherwise, the value of the Response-PDU's error-status field is set to
`noError', and the value of its error-index field to zero.

The generated Response-PDU (possibly with an empty variable-bindings
field) is then encapsulated into a message. If the size of the
resultant message is less than or equal to both a local constraint and
the maximum message size of the originator, it is transmitted to the
originator of the GetBulkRequest-PDU. Otherwise, the snmpSilentDrops |
[10] counter is incremented and the resultant message is discarded.

4.2.3.1. **Another Example of Table Traversal**

This example demonstrates how the GetBulkRequest-PDU can be used as an
alternative to the GetNextRequest-PDU. The same traversal of the IP
net-to-media table as shown in Section 4.2.2.1 is achieved with fewer
exchanges.

The SNMPv2 entity acting in a manager role begins by sending a
GetBulkRequest-PDU with the modest max-repetitions value of 2, and
containing the indicated OBJECT IDENTIFIER values as the requested
variable names:

    GetBulkRequest [non-repeaters = 1, max-repetitions = 2]
                   (sysUpTime,
                     ipNetToMediaPhysAddress,
                     ipNetToMediaType)

The SNMPv2 entity acting in an agent role responds with a Response-PDU:

```
Response (( sysUpTime.0 =  "123456" ),
          ( ipNetToMediaPhysAddress.1.9.2.3.4 =
                                  "000010543210" ),
          ( ipNetToMediaType.1.9.2.3.4 =  "dynamic" ),
          ( ipNetToMediaPhysAddress.1.10.0.0.51 =
                                  "000010012345" ),
          ( ipNetToMediaType.1.10.0.0.51 =  "static" ))
```

The SNMPv2 entity acting in a manager role continues with:

```
GetBulkRequest [ non-repeaters = 1, max-repetitions = 2 ]
               ( sysUpTime,
                 ipNetToMediaPhysAddress.1.10.0.0.51,
                 ipNetToMediaType.1.10.0.0.51 )
```

The SNMPv2 entity acting in an agent role responds with:

```
Response (( sysUpTime.0 =  "123466" ),
          ( ipNetToMediaPhysAddress.2.10.0.0.15 =
                                  "000010987654" ),
          ( ipNetToMediaType.2.10.0.0.15 =
                                      "dynamic" ),
          ( ipNetToMediaNetAddress.1.9.2.3.4 =
                                      "9.2.3.4" ),
          ( ipRoutingDiscards.0 =  "2" ))
```

This response signals the end of the table to the SNMPv2 entity acting in a manager role.

4.2.4. The Response-PDU

The Response-PDU is generated by a SNMPv2 entity only upon receipt of a GetRequest-PDU, GetNextRequest-PDU, GetBulkRequest-PDU, SetRequest-PDU, or InformRequest-PDU, as described elsewhere in this document.

If the error-status field of the Response-PDU is non-zero, the value fields of the variable bindings in the variable binding list are ignored.

If both the error-status field and the error-index field of the Response-PDU are non-zero, then the value of the error-index field is the index of the variable binding (in the variable-binding list of the

corresponding request) for which the request failed. The first variable
binding in a request's variable-binding list is index one, the second is
index two, etc.

A compliant SNMPv2 entity acting in a manager role must be able to
properly receive and handle a Response-PDU with an error-status field
equal to `noSuchName', `badValue', or `readOnly'. (See Section 3.1.2 of
[9].)

Upon receipt of a Response-PDU, the receiving SNMPv2 entity presents its
contents to the SNMPv2 application which generated the request with the
same request-id value.

4.2.5. **The SetRequest-PDU**

A SetRequest-PDU is generated and transmitted at the request of a SNMPv2
application.

Upon receipt of a SetRequest-PDU, the receiving SNMPv2 entity determines
the size of a message encapsulating a Response-PDU having the same
values in its request-id and variable-bindings fields as the received
SetRequest-PDU, and the largest possible sizes of the error-status and
error-index fields. If the determined message size is greater than
either a local constraint or the maximum message size of the originator,
then an alternate Response-PDU is generated, transmitted to the
originator of the SetRequest-PDU, and processing of the SetRequest-PDU
terminates immediately thereafter. This alternate Response-PDU is
formatted with the same values in its request-id field as the received
SetRequest-PDU, with the value of its error-status field set to
`tooBig', the value of its error-index field set to zero, and an empty
variable-bindings field. This alternate Response-PDU is then
encapsulated into a message. If the size of the resultant message is
less than or equal to both a local constraint and the maximum message
size of the originator, it is transmitted to the originator of the
SetRequest-PDU. Otherwise, the snmpSilentDrops |
[10] counter is incremented and the resultant message is discarded.
Regardless, processing of the SetRequest-PDU terminates.

Otherwise, the receiving SNMPv2 entity processes each variable binding
in the variable-binding list to produce a Response-PDU. All fields of
the Response-PDU have the same values as the corresponding fields of the
received request except as indicated below.

The variable bindings are conceptually processed as a two phase
operation. In the first phase, each variable binding is validated; if
all validations are successful, then each variable is altered in the
second phase. Of course, implementors are at liberty to implement
either the first, or second, or both, of these conceptual phases as
multiple implementation phases. Indeed, such multiple implementation
phases may be necessary in some cases to ensure consistency.

The following validations are performed in the first phase on each
variable binding until they are all successful, or until one fails:

(1) If the variable binding's name specifies an existing or non-
     existent variable to which this request is/would be denied access
     because it is/would not be in the appropriate MIB view, then the
     value of the Response-PDU's error-status field is set to
     `noAccess', and the value of its error-index field is set to the
     index of the failed variable binding.

(2) Otherwise, if there are no variables which share the same OBJECT
     IDENTIFIER prefix as the variable binding's name, and which are
     able to be created or modified no matter what new value is
     specified, then the value of the Response-PDU's error-status field
     is set to `notWritable', and the value of its error-index field is
     set to the index of the failed variable binding.

(3) Otherwise, if the variable binding's value field specifies,
     according to the ASN.1 language, a type which is inconsistent with
     that required for all variables which share the same OBJECT
     IDENTIFIER prefix as the variable binding's name, then the value of
     the Response-PDU's error-status field is set to `wrongType', and
     the value of its error-index field is set to the index of the
     failed variable binding.

(4) Otherwise, if the variable binding's value field specifies,
     according to the ASN.1 language, a length which is inconsistent
     with that required for all variables which share the same OBJECT
     IDENTIFIER prefix as the variable binding's name, then the value of
     the Response-PDU's error-status field is set to `wrongLength', and
     the value of its error-index field is set to the index of the
     failed variable binding.

(5) Otherwise, if the variable binding's value field contains an ASN.1
     encoding which is inconsistent with that field's ASN.1 tag, then
     the value of the Response-PDU's error-status field is set to
     `wrongEncoding', and the value of its error-index field is set to

the index of the failed variable binding. (Note that not all
implementation strategies will generate this error.)

(6) Otherwise, if the variable binding's value field specifies a value
which could under no circumstances be assigned to the variable,
then the value of the Response-PDU's error-status field is set to
`wrongValue', and the value of its error-index field is set to the
index of the failed variable binding.

(7) Otherwise, if the variable binding's name specifies a variable
which does not exist and could not ever be created (even though
some variables sharing the same OBJECT IDENTIFIER prefix might
under some circumstances be able to be created), then the value of
the Response-PDU's error-status field is set to `noCreation', and
the value of its error-index field is set to the index of the
failed variable binding.

(8) Otherwise, if the variable binding's name specifies a variable
which does not exist but can not be created under the present
circumstances (even though it could be created under other
circumstances), then the value of the Response-PDU's error-status
field is set to `inconsistentName', and the value of its error-
index field is set to the index of the failed variable binding.

(9) Otherwise, if the variable binding's name specifies a variable
which exists but can not be modified no matter what new value is
specified, then the value of the Response-PDU's error-status field
is set to `notWritable', and the value of its error-index field is
set to the index of the failed variable binding.

(10) Otherwise, if the variable binding's value field specifies a value
that could under other circumstances be held by the variable, but
is presently inconsistent or otherwise unable to be assigned to the
variable, then the value of the Response-PDU's error-status field
is set to `inconsistentValue', and the value of its error-index
field is set to the index of the failed variable binding.

(11) When, during the above steps, the assignment of the value specified
by the variable binding's value field to the specified variable
requires the allocation of a resource which is presently
unavailable, then the value of the Response-PDU's error-status
field is set to `resourceUnavailable', and the value of its error-
index field is set to the index of the failed variable binding.

(12) If the processing of the variable binding fails for a reason other
     than listed above, then the value of the Response-PDU's error-
     status field is set to `genErr', and the value of its error-index
     field is set to the index of the failed variable binding.

(13) Otherwise, the validation of the variable binding succeeds.

At the end of the first phase, if the validation of all variable
bindings succeeded, then the value of the Response-PDU's error-status
field is set to `noError' and the value of its error-index field is
zero, and processing continues as follows.

For each variable binding in the request, the named variable is created
if necessary, and the specified value is assigned to it. Each of these
variable assignments occurs as if simultaneously with respect to all
other assignments specified in the same request. However, if the same
variable is named more than once in a single request, with different
associated values, then the actual assignment made to that variable is
implementation-specific.

If any of these assignments fail (even after all the previous
validations), then all other assignments are undone, and the Response-
PDU is modified to have the value of its error-status field set to
`commitFailed', and the value of its error-index field set to the index
of the failed variable binding.

If and only if it is not possible to undo all the assignments, then the
Response-PDU is modified to have the value of its error-status field set
to `undoFailed', and the value of its error-index field is set to zero.
Note that implementations are strongly encouraged to take all possible
measures to avoid use of either `commitFailed' or `undoFailed' - these
two error-status codes are not to be taken as license to take the easy
way out in an implementation.

Finally, the generated Response-PDU is encapsulated into a message, and
transmitted to the originator of the SetRequest-PDU.

4.2.6. **The SNMPv2-Trap-PDU**

A SNMPv2-Trap-PDU is generated and transmitted by a SNMPv2 entity acting
in an agent role when an exceptional situation occurs.

The destination(s) to which a SNMPv2-Trap-PDU is sent is determined in
an implementation-dependent fashion by the SNMPv2 entity. The first two

variable bindings in the variable binding list of an SNMPv2-Trap-PDU are
sysUpTime.0 [10] and snmpTrapOID.0 [10] respectively. If the OBJECTS
clause is present in the invocation of the corresponding NOTIFICATION-
TYPE macro, then each corresponding variable, as instantiated by this
notification, is copied, in order, to the variable-bindings field. If
any additional variables are being included (at the option of the
generating SNMPv2 entity), then each is copied to the variable-bindings
field.

4.2.7. The InformRequest-PDU

An InformRequest-PDU is generated and transmitted at the request of an
application in a SNMPv2 entity acting in a manager role, that wishes to
notify another application (in a SNMPv2 entity also acting in a manager
role) of information in a MIB view which is remote to the receiving
application.

The destination(s) to which an InformRequest-PDU is sent is specified by
the requesting application. The first two variable bindings in the
variable binding list of an InformRequest-PDU are sysUpTime.0 [10] and
snmpTrapOID.0 [10] respectively. If the OBJECTS clause is present in
the invocation of the corresponding NOTIFICATION-TYPE macro, then each
corresponding variable, as instantiated by this notification, is copied,
in order, to the variable-bindings field.

Upon receipt of an InformRequest-PDU, the receiving SNMPv2 entity
determines the size of a message encapsulating a Response-PDU with the
same values in its request-id, error-status, error-index and variable-
bindings fields as the received InformRequest-PDU. If the determined
message size is greater than either a local constraint or the maximum
message size of the originator, then an alternate Response-PDU is
generated, transmitted to the originator of the InformRequest-PDU, and
processing of the InformRequest-PDU terminates immediately thereafter.
This alternate Response-PDU is formatted with the same values in its
request-id field as the received InformRequest-PDU, with the value of
its error-status field set to `tooBig', the value of its error-index
field set to zero, and an empty variable-bindings field. This alternate
Response-PDU is then encapsulated into a message. If the size of the
resultant message is less than or equal to both a local constraint and
the maximum message size of the originator, it is transmitted to the
originator of the InformRequest-PDU. Otherwise, the snmpSilentDrops |
[10] counter is incremented and the resultant message is discarded.
Regardless, processing of the InformRequest-PDU terminates.

Otherwise, the receiving SNMPv2 entity:

(1) presents its contents to the appropriate SNMPv2 application;

(2) generates a Response-PDU with the same values in its request-id and
     variable-bindings fields as the received InformRequest-PDU, with
     the value of its error-status field is set to `noError' and the
     value of its error-index field is zero; and

(3) transmits the generated Response-PDU to the originator of the
     InformRequest-PDU.

5. Acknowledgements

This document is the result of significant work by the four major
contributors:

    Jeffrey Case (SNMP Research, case@snmp.com)
    Keith McCloghrie (Cisco Systems, kzm@cisco.com)
    Marshall Rose (Dover Beach Consulting, mrose@dbc.mtview.ca.us)
    Steven Waldbusser (International Network Services, stevew@uni.ins.com)

In addition, the contributions of the SNMPv2 Working Group are
acknowledged. In particular, a special thanks is extended for the
contributions of:

    Alexander I. Alten (Novell)
    Dave Arneson (Cabletron)
    Uri Blumenthal (IBM)
    Doug Book (Chipcom)
    Kim Curran (Bell-Northern Research)
    Jim Galvin (Trusted Information Systems)
    Maria Greene (Ascom Timeplex)
    Iain Hanson (Digital)
    Dave Harrington (Cabletron)
    Nguyen Hien (IBM)
    Jeff Johnson (Cisco Systems)
    Michael Kornegay (Object Quest)
    Deirdre Kostick (AT&T Bell Labs)
    David Levi (SNMP Research)
    Daniel Mahoney (Cabletron)
    Bob Natale (ACE*COMM)
    Brian O'Keefe (Hewlett Packard)
    Andrew Pearson (SNMP Research)
    Dave Perkins (Peer Networks)
    Randy Presuhn (Peer Networks)
    Aleksey Romanov (Quality Quorum)
    Shawn Routhier (Epilogue)
    Jon Saperia (BGS Systems)
    Bob Stewart (Cisco Systems, bstewart@cisco.com), chair
    Kaj Tesink (Bellcore)
    Glenn Waters (Bell-Northern Research)
    Bert Wijnen (IBM)

6. References

[1]   Information processing systems - Open Systems Interconnection -
      Specification of Abstract Syntax Notation One (ASN.1),
      International Organization for Standardization. International
      Standard 8824, (December, 1987).

[2]   McCloghrie, K., Editor, |
      "Structure of Management Information for Version 2 of the Simple
      Network Management Protocol (SNMPv2)", Internet Draft, Cisco |
      Systems, September 1995. |

[3]   McCloghrie, K., Editor, |
      "Introduction to Version 2 of the Internet-standard Network
      Management Framework", Internet Draft, Cisco Systems, September |
      1995. |

[4]   McCloghrie, K., Editor, |
      "Textual Conventions for Version 2 of the Simple Network Management
      Protocol (SNMPv2)", Internet Draft, Cisco Systems, September 1995. |

[5]   C. Kent, J. Mogul, Fragmentation Considered Harmful, Proceedings,
      ACM SIGCOMM '87, Stowe, VT, (August 1987).

[6]   McCloghrie, K., Editor, |
      "Transport Mappings for Version 2 of the Simple Network Management
      Protocol (SNMPv2)", Internet Draft, Cisco Systems, September 1995. |

[7]   Postel, J., "User Datagram Protocol", STD 6, RFC 768,
      USC/Information Sciences Institute, August 1980.

[8]   McCloghrie, K., Editor, |
      "SNMPv2 Management Information Base for the Internet Protocol",
      Internet Draft, Cisco Systems, September 1995. |

[9]   McCloghrie, K., Editor, |
      "Coexistence between Version 1 and Version 2 of the Internet-
      standard Network Management Framework", Internet Draft, Cisco |
      Systems, September 1995. |

[10] McCloghrie, K., Editor, |
      "Management Information Base for Version 2 of the Simple Network
      Management Protocol (SNMPv2)", Internet Draft, Cisco Systems, |
      September 1995. |

7. **Security Considerations**

Security issues are not discussed in this memo.

8. **Editor's Address**

   Keith McCloghrie   -
   Cisco Systems, Inc.
   170 West Tasman Drive
   San Jose, CA 95134-1706
   US

   Phone: +1 408 526 5260
   Email: kzm@cisco.com

Table of Contents   -