Network Working Group
Request for Comments: 1353

K. McCloghrie
Hughes LAN Systems, Inc.
J. Davin
MIT Laboratory for Computer Science
J. Galvin
Trusted Information Systems, Inc.

July 1992

Definitions of Managed Objects for Administration of SNMP Parties

Status of this Memo

This document specifies an IAB standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "IAB Official Protocol Standards" for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in TCP/IP-based internets. In particular, it describes a representation of the SNMP parties defined in [8] as objects defined according to the Internet Standard SMI [1]. These definitions are consistent with the SNMP Security protocols set forth in [9].

Table of Contents

1. The Network Management Framework

the Internet-standard Network Management Framework consists of three components. They are:

RFC 1155 which defines the SMI, the mechanisms used for describing and naming objects for the purpose of management. RFC 1212 defines a more concise description mechanism, which is wholly consistent with the SMI.

RFC 1156 which defines MIB-I, the core set of managed objects for the Internet suite of protocols. RFC 1213, defines MIB-II, an evolution of MIB-I based on implementation experience and new operational requirements.

RFC 1157 which defines the SNMP, the protocol used for network access to managed objects.

The Framework permits new objects to be defined for the purpose of experimentation and evaluation.

2. Objects

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the subset of Abstract Syntax Notation One (ASN.1) [5] defined in the SMI. In particular, each object has a name, a syntax, and an encoding. The name is an object identifier, an administratively assigned name, which specifies an object type. The object type together with an object instance serves to uniquely identify a specific instantiation of the object. For human convenience, we often use a textual string, termed the OBJECT DESCRIPTOR, to also refer to the object type.

The syntax of an object type defines the abstract data structure corresponding to that object type. The ASN.1 language is used for this purpose. However, the SMI [1] purposely restricts the ASN.1 constructs which may be used. These restrictions are explicitly made for simplicity.

The encoding of an object type is simply how that object type is represented using the object type’s syntax. Implicitly tied to the notion of an object type’s syntax and encoding is how the object type is represented when being transmitted on the network.

The SMI specifies the use of the basic encoding rules of ASN.1 [6], subject to the additional requirements imposed by the SNMP.

2.1. Format of Definitions

Section 4 contains the specification of all object types contained in this MIB module. The object types are defined using the conventions defined in the SMI, as amended by the extensions specified in [7].

3. Overview

3.1. Structure

This MIB contains the definitions for four tables, a number of OBJECT IDENTIFIER assignments, and some conventions for initial use with some of the assignments. The four tables are the SNMP Party Public database, the SNMP Party Secrets database, the SNMP Access Control database, and the SNMP Views database.

The SNMP Party Public database and the SNMP Party Secrets database are defined as separate tables specifically for the purpose of positioning them in different parts of the MIB tree namespace. In particular, the SNMP Party Secrets database contains secret information, for which security demands that access to it be limited to parties which use both authentication and privacy. It is therefore positioned in a separate branch of the MIB tree so as to provide for the easiest means of accommodating the required limitation.

In contrast, the SNMP Party Public database contains public information about SNMP parties. In particular, it contains the parties' clocks which need to be read-able (but not write-able) by unauthenticated queries, since an unauthenticated query of a party's clock is the first step of the procedure to re-establish clock synchronization (see [9]).

The objects in this MIB are organized into four groups. All four of the groups are mandatory for those SNMP implementations that realize the security framework and mechanisms defined in [8] and [9].

3.2. Instance Identifiers

In all four of the tables in this MIB, the object instances are identified by values which have an underlying syntax of OBJECT IDENTIFIER. For the Party Public database and the Party Secrets database, the index variable is the party identifier. For the Access Control database and the Views database, two index variables are defined, both of which have a syntax of OBJECT IDENTIFIER. (See the INDEX clauses in the MIB definitions below for the specific variables.)

According to RFC 1212 [7], section 4.1.6, the syntax of the object(s) specified in an INDEX clause indicates how to form the instance-identifier. In particular, for each index object which is object identifier-valued, its contribution to the instance identifier is:

> 'n+1' sub-identifiers, where 'n' is the number of sub-identifiers in the value (the first sub-identifier is 'n' itself, following this, each sub-identifier in the value is copied).

3.3. Textual Conventions

The datatypes, Party, Clock, and TAddress, are used as textual conventions in this document. These textual conventions have NO effect on either the syntax nor the semantics of any managed object. Objects defined using these conventions are always encoded by means of the rules that define their primitive type. Hence, no changes to the SMI or the SNMP are necessary to accommodate these textual conventions which are adopted merely for the convenience of readers.

4. Definitions

```
RFC1353-MIB DEFINITIONS ::= BEGIN

IMPORTS
        system, mib, private, internet    FROM RFC1155-SMI
        OBJECT-TYPE                       FROM RFC-1212;

snmpParties     OBJECT IDENTIFIER ::= { mib-2 20 }
partyAdmin      OBJECT IDENTIFIER ::= { snmpParties 1 }
partyPublic     OBJECT IDENTIFIER ::= { snmpParties 2 }

snmpSecrets     OBJECT IDENTIFIER ::= { mib-2 21 }
partyPrivate    OBJECT IDENTIFIER ::= { snmpSecrets 1 }
partyAccess     OBJECT IDENTIFIER ::= { snmpSecrets 2 }
partyViews      OBJECT IDENTIFIER ::= { snmpSecrets 3 }


--                  Textual Conventions

--    A textual convention denoting a SNMP party identifier:

Party ::= OBJECT IDENTIFIER


--    A party's authentication clock - a non-negative integer
-- which is incremented as specified/allowed by the party's
-- Authentication Protocol.
--    For noAuth, a party's authentication clock is unused and
```

```
-- its value is undefined.
--    For md5AuthProtocol, a party's authentication clock is a
-- relative clock with 1-second granularity.

Clock ::= INTEGER (0..2147483647)


--    A textual convention denoting a transport service
-- address.
--    For rfc1351Domain, a TAddress is 6 octets long,
-- the initial 4 octets containing the IP-address in
-- network-byte order and the last 2 containing the
-- UDP port in network-byte order.

TAddress ::= OCTET STRING


--- Definitions of Security Protocols

partyProtocols
    OBJECT IDENTIFIER ::= { partyAdmin 1 }

noAuth                  -- The protocol without authentication
    OBJECT IDENTIFIER ::= { partyProtocols 1 }

noPriv                  -- The protocol without privacy
    OBJECT IDENTIFIER ::= { partyProtocols 3 }

desPrivProtocol         -- The DES Privacy Protocol
    OBJECT IDENTIFIER ::= { partyProtocols 4 }

md5AuthProtocol         -- The MD5 Authentication Protocol
    OBJECT IDENTIFIER ::= { partyProtocols 5 }


--- definitions of Transport Domains

transportDomains
    OBJECT IDENTIFIER ::= { partyAdmin 2 }

rfc1351Domain --- RFC-1351 (SNMP over UDP, using SNMP Parties)
    OBJECT IDENTIFIER ::= { transportDomains 1 }
```

```
--- definitions of Proxy Domains

proxyDomains
    OBJECT IDENTIFIER ::= { partyAdmin 3 }

noProxy                 --- Local operation
    OBJECT IDENTIFIER ::= { proxyDomains 1 }


---    Definition of Initial Party Identifiers

--  When devices are installed, they need to be configured
--  with an initial set of SNMP parties.  The configuration
--  of SNMP parties requires (among other things) the
--  assignment of several OBJECT IDENTIFIERs.  Any local
--  network administration can obtain the delegated
--  authority necessary to assign its own OBJECT
--  IDENTIFIERs.  However, to provide for those
--  administrations who have not obtained the necessary
--  authority, this document allocates a branch of the
--  naming tree for use with the following conventions.

initialPartyId
    OBJECT IDENTIFIER ::= { partyAdmin 4 }

--  Note these are identified as "initial" party identifiers
--  since these allow secure SNMP communication to proceed,
--  thereby allowing further SNMP parties to be configured
--  through use of the SNMP itself.

--  The following definitions identify a party identifier,
--  and specify the initial values of various object
--  instances indexed by that identifier.  In addition,
--  the initial MIB view and access control parameters
--  assigned, by convention, to these parties are identified.

--    Party Identifiers for use as initial SNMP parties
--       at IP address  a.b.c.d

-- partyIdentity            = { initialPartyId a b c d 1 }
-- partyTDomain             = { rfc1351Domain }
-- partyTAddress            = a.b.c.d, 161
-- partyProxyFor            = { noProxy }
-- partyAuthProtocol        = { noAuth }
-- partyAuthClock           = 0
-- partySecretsAuthPrivate  = ''h    (the empty string)
-- partyAuthPublic          = ''h    (the empty string)
-- partyAuthLifetime        = 0
```

```
        -- partyPrivProtocol        = { noPriv }
        -- partySecretsPrivPrivate  = ''h    (the empty string)
        -- partyPrivPublic          = ''h    (the empty string)

        -- partyIdentity            = { initialPartyId a b c d 2 }
        -- partyTDomain             = { rfc1351Domain }
        -- partyTAddress            = assigned by local administration
        -- partyProxyFor            = { noProxy }
        -- partyAuthProtocol        = { noAuth }
        -- partyAuthClock           = 0
        -- partySecretsAuthPrivate  = ''h    (the empty string)
        -- partyAuthPublic          = ''h    (the empty string)
        -- partyAuthLifetime        = 0
        -- partyPrivProtocol        = { noPriv }
        -- partySecretsPrivPrivate  = ''h    (the empty string)
        -- partyPrivPublic          = ''h    (the empty string)

        -- partyIdentity            = { initialPartyId a b c d 3 }
        -- partyTDomain             = { rfc1351Domain }
        -- partyTAddress            = a.b.c.d, 161
        -- partyProxyFor            = { noProxy }
        -- partyAuthProtocol        = { md5AuthProtocol }
        -- partyAuthClock           = 0
        -- partySecretsAuthPrivate  = assigned by local administration
        -- partyAuthPublic          = ''h    (the empty string)
        -- partyAuthLifetime        = 300
        -- partyPrivProtocol        = { noPriv }
        -- partySecretsPrivPrivate  = ''h    (the empty string)
        -- partyPrivPublic          = ''h    (the empty string)

        -- partyIdentity            = { initialPartyId a b c d 4 }
        -- partyTDomain             = { rfc1351Domain }
        -- partyTAddress            = assigned by local administration
        -- partyProxyFor            = { noProxy }
        -- partyAuthProtocol        = { md5AuthProtocol }
        -- partyAuthClock           = 0
        -- partySecretsAuthPrivate  = assigned by local administration
        -- partyAuthPublic          = ''h    (the empty string)
        -- partyAuthLifetime        = 300
        -- partyPrivProtocol        = { noPriv }
        -- partySecretsPrivPrivate  = ''h    (the empty string)
        -- partyPrivPublic          = ''h    (the empty string)

        -- partyIdentity            = { initialPartyId a b c d 5 }
        -- partyTDomain             = { rfc1351Domain }
        -- partyTAddress            = a.b.c.d, 161
        -- partyProxyFor            = { noProxy }
        -- partyAuthProtocol        = { md5AuthProtocol }
```

```
-- partyAuthClock           = 0
-- partySecretsAuthPrivate  = assigned by local administration
-- partyAuthPublic          = ''h    (the empty string)
-- partyAuthLifetime        = 300
-- partyPrivProtocol        = { desPrivProtocol }
-- partySecretsPrivPrivate  = assigned by local administration
-- partyPrivPublic          = ''h    (the empty string)

-- partyIdentity            = { initialPartyId a b c d 6 }
-- partyTDomain             = { rfc1351Domain }
-- partyTAddress            = assigned by local administration
-- partyProxyFor            = { noProxy }
-- partyAuthProtocol        = { md5AuthProtocol }
-- partyAuthClock           = 0
-- partySecretsAuthPrivate  = assigned by local administration
-- partyAuthPublic          = ''h    (the empty string)
-- partyAuthLifetime        = 300
-- partyPrivProtocol        = { desPrivProtocol }
-- partySecretsPrivPrivate  = assigned by local administration
-- partyPrivPublic          = ''h   (the empty string)


--  The initial access control parameters assigned, by
--  convention, to these parties are:

-- aclTarget     = { initialPartyId a b c d 1 }
-- aclSubject    = { initialPartyId a b c d 2 }
-- aclPrivileges = 3 (Get & Get-Next)

-- aclTarget     = { initialPartyId a b c d 2 }
-- aclSubject    = { initialPartyId a b c d 1 }
-- aclPrivileges = 20 (GetResponse & Trap)

-- aclTarget     = { initialPartyId a b c d 3 }
-- aclSubject    = { initialPartyId a b c d 4 }
-- aclPrivileges = 11 (Get, Get-Next & Set)

-- aclTarget     = { initialPartyId a b c d 4 }
-- aclSubject    = { initialPartyId a b c d 3 }
-- aclPrivileges = 20 (GetResponse & Trap)

-- aclTarget     = { initialPartyId a b c d 5 }
-- aclSubject    = { initialPartyId a b c d 6 }
-- aclPrivileges = 11 (Get, Get-Next & Set)

-- aclTarget     = { initialPartyId a b c d 6 }
-- aclSubject    = { initialPartyId a b c d 5 }
-- aclPrivileges = 20 (GetResponse & Trap)
```

```
        --  The initial MIB views assigned, by convention, to
        --  these parties are:

        -- viewParty    = { initialPartyId a b c d 1 }
        -- viewSubtree  = { system }
        -- viewStatus   = { included }
        -- viewMask     = { ''h }

        -- viewParty    = { initialPartyId a b c d 1 }
        -- viewSubtree  = { snmpParties }
        -- viewStatus   = { included }
        -- viewMask     = { ''h }

        -- viewParty    = { initialPartyId a b c d 3 }
        -- viewSubtree  = { internet }
        -- viewStatus   = { included }
        -- viewMask     = { ''h }

        -- viewParty    = { initialPartyId a b c d 3 }
        -- viewSubtree  = { partyPrivate }
        -- viewStatus   = { excluded }
        -- viewMask     = { ''h }

        -- viewParty    = { initialPartyId a b c d 5 }
        -- viewSubtree  = { internet }
        -- viewStatus   = { included }
        -- viewMask     = { ''h }


        --   The SNMP Party Public Database Group
        --
        -- The non-secret party information.
        --
        -- Implementation of the objects in this group is mandatory.

        partyTable OBJECT-TYPE
            SYNTAX  SEQUENCE OF PartyEntry
            ACCESS  not-accessible
            STATUS  mandatory
            DESCRIPTION
                    "The SNMP Party Public database.

                    An agent must ensure that there is, at all times,
                    a one-to-one correspondence between entries in
                    this table and entries in the partySecretsTable.

                    The creation/deletion of instances in this table
                    via SNMP Set-Requests is not allowed.  Instead,
```

```
                    entries in this table are created/deleted as a
                    side-effect of the creation/deletion of
                    corresponding entries in the partySecretsTable.

                    Thus, a SNMP Set-Request whose varbinds contain a
                    reference to a non-existent instance of a
                    partyTable object, but no reference to the
                    corresponding instance of a partySecretsTable
                    object, will be rejected."
            ::= { partyPublic 1 }

            partyEntry OBJECT-TYPE
                SYNTAX  PartyEntry
                ACCESS  not-accessible
                STATUS  mandatory
                DESCRIPTION
                        "Locally held non-secret information about a
                        particular SNMP party, which is available for
                        access by network management.  Note that this does
                        not include all locally held information about a
                        party.  In particular, it does not include the
                        'last-timestamp' (i.e., the timestamp of the last
                        authentic message received) or the 'nonce'
                        values."
                INDEX  { partyIdentity }
                ::= { partyTable 1 }

            PartyEntry ::=
                SEQUENCE {
                    partyIdentity
                        Party,
                    partyTDomain
                        OBJECT IDENTIFIER,
                    partyTAddress
                        TAddress,
                    partyProxyFor
                        Party,
                    partyAuthProtocol
                        OBJECT IDENTIFIER,
                    partyAuthClock
                        Clock,
                    partyAuthPublic
                        OCTET STRING,
                    partyAuthLifetime
                        INTEGER,
                    partyPrivProtocol
                        OBJECT IDENTIFIER,
                    partyPrivPublic
```

```
                OCTET STRING,
            partyMaxMessageSize
                INTEGER,
            partyStatus
                INTEGER
        }

    partyIdentity  OBJECT-TYPE
        SYNTAX  Party
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "A party identifier uniquely identifying a
                particular SNMP party."
        ::= { partyEntry 1 }

    partyTDomain  OBJECT-TYPE
        SYNTAX  OBJECT IDENTIFIER
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "Indicates the kind of transport service by which
                the party receives network management traffic. An
                example of a transport domain is 'rfc1351Domain'
                (SNMP over UDP)."
        DEFVAL  { rfc1351Domain }
        ::= { partyEntry 2 }

    partyTAddress  OBJECT-TYPE
        SYNTAX  TAddress
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The transport service address by which the party
                receives network management traffic, formatted
                according to the corresponding value of
                partyTDomain.  For rfc1351Domain, partyTAddress is
                formatted as a 4-octet IP Address concatenated
                with a 2-octet UDP port number."
        DEFVAL  { '000000000000'h }
        ::= { partyEntry 3 }

    partyProxyFor OBJECT-TYPE
        SYNTAX  Party
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The identity of a second SNMP party or other
```

```
                management entity with which interaction may be
                necessary to satisfy received management requests.
                In this context, the distinguished value { noProxy
                } signifies that the party responds to received
                management requests by entirely local mechanisms."
        DEFVAL  { noProxy }
        ::= { partyEntry 4 }

    partyAuthProtocol OBJECT-TYPE
        SYNTAX  OBJECT IDENTIFIER
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The authentication protocol by which all messages
                generated by the party are authenticated as to
                origin and integrity.  In this context, the value
                { noAuth } signifies that messages generated by
                the party are not authenticated."
        DEFVAL  { md5AuthProtocol }
        ::= { partyEntry 5 }

    partyAuthClock OBJECT-TYPE
        SYNTAX  Clock
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The authentication clock which represents the
                local notion of the current time specific to the
                party.  This value must not be decremented unless
                the party's secret information is changed
                simultaneously, at which time the party's nonce
                and last-timestamp values must also be reset to
                zero, and the new value of the clock,
                respectively."
        DEFVAL  { 0 }
        ::= { partyEntry 6 }

    partyAuthPublic OBJECT-TYPE
        SYNTAX  OCTET STRING -- for md5AuthProtocol: (SIZE (0..16))
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "A publically-readable value for the party.

                Depending on the party's authentication protocol,
                this value may be needed to support the party's
                authentication protocol.  Alternatively, it may be
                used by a manager during the procedure for
```

```
                altering secret information about a party.  (For
                example, by altering the value of an instance of
                this object in the same SNMP Set-Request used to
                update an instance of partyAuthPrivate, a
                subsequent Get-Request can determine if the Set-
                Request was successful in the event that no
                response to the Set-Request is received, see RFC
                1352.)

                The length of the value is dependent on the
                party's authentication protocol.  If not used by
                the authentication protocol, it is recommended
                that agents support values of any length up to and
                including the length of the corresponding
                partyAuthPrivate object."
        DEFVAL  { ''h }      -- the empty string
        ::= { partyEntry 7 }

    partyAuthLifetime OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The lifetime (in units of seconds) which
                represents an administrative upper bound on
                acceptable delivery delay for protocol messages
                generated by the party."
        DEFVAL  { 300 }
        ::= { partyEntry 8 }

    partyPrivProtocol OBJECT-TYPE
        SYNTAX  OBJECT IDENTIFIER
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The privacy protocol by which all protocol
                messages received by the party are protected from
                disclosure.  In this context, the value { noPriv }
                signifies that messages received by the party are
                not protected."
        DEFVAL  { noPriv }
        ::= { partyEntry 9 }

    partyPrivPublic OBJECT-TYPE
        SYNTAX  OCTET STRING -- for desPrivProtocol: (SIZE (0..16))
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
```

```
                    "A publically-readable value for the party.

                    Depending on the party's privacy protocol, this
                    value may be needed to support the party's privacy
                    protocol.  Alternatively, it may be used by a
                    manager as a part of its procedure for altering
                    secret information about a party.  (For example,
                    by altering the value of an instance of this
                    object in the same SNMP Set-Request used to update
                    an instance of partyPrivPrivate, a subsequent
                    Get-Request can determine if the Set-Request was
                    successful in the event that no response to the
                    Set-Request is received, see RFC 1352.)

                    The length of the value is dependent on the
                    party's privacy protocol.  If not used by the
                    privacy protocol, it is recommended that agents
                    support values of any length up to and including
                    the length of the corresponding partyPrivPrivate
                    object."
            DEFVAL  { ''h }     -- the empty string
            ::= { partyEntry 10 }

        partyMaxMessageSize OBJECT-TYPE
            SYNTAX  INTEGER (484..65507)
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "The maximum length in octets of a SNMP message
                    which this party will accept.  For parties which
                    execute at an agent, the agent initializes this
                    object to the maximum length supported by the
                    agent, and does not let the object be set to any
                    larger value.  For parties which do not execute at
                    the agent, the agent must allow the manager to set
                    this object to any legal value, even if it is
                    larger than the agent can generate."
            DEFVAL  { 484 }
            ::= { partyEntry 11 }

        partyStatus OBJECT-TYPE
            SYNTAX  INTEGER  { valid(1), invalid(2) }
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "The status of the locally-held information on a
                    particular SNMP party.
```

```
                The instance of this object for a particular party
                and the instance of partySecretsStatus for the
                same party always have the same value.

                This object will typically provide unrestricted
                read-only access to the status of parties.  In
                contrast, partySecretsStatus will typically
                provide restricted read-write access to the status
                of parties."
        ::= { partyEntry 12 }


--   The SNMP Party Secrets Database Group

-- The secret party information
--
-- Implementation of the objects in this group is mandatory.

partySecretsTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF PartySecretsEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "The SNMP Party Secrets database."
::= { partyPrivate 1 }

partySecretsEntry OBJECT-TYPE
    SYNTAX  PartySecretsEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "Locally held secret information about a
            particular SNMP party, which is available for
            access by network management.

            When a SNMP Set-Request is used to update the
            values of instances of objects in this table, it
            is recommended that the same SNMP Set-Request also
            alter the value of a non-secret object instance
            (e.g., an instance of partyAuthPublic or
            partyPrivPublic).  This allows a Get-Request of
            that non-secret object instance to determine if
            the Set-Request was successful in the event that
            no response which matches the Set-Request is
            received, see RFC 1352."
    INDEX  { partySecretsIdentity }
    ::= { partySecretsTable 1 }
```

```
        PartySecretsEntry ::=
            SEQUENCE {
                partySecretsIdentity
                    Party,
                partySecretsAuthPrivate
                    OCTET STRING,
                partySecretsPrivPrivate
                    OCTET STRING,
                partySecretsStatus
                    INTEGER
            }

        partySecretsIdentity  OBJECT-TYPE
            SYNTAX  Party
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "A party identifier uniquely identifying a
                    particular SNMP party."
            ::= { partySecretsEntry 1 }

        partySecretsAuthPrivate OBJECT-TYPE
            SYNTAX  OCTET STRING   -- for md5AuthProtocol: (SIZE (16))
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "An encoding of the party's private authentication
                    key which may be needed to support the
                    authentication protocol.  Although the value of
                    this variable may be altered by a management
                    operation (e.g., a SNMP Set-Request), its value
                    can never be retrieved by a management operation:
                    when read, the value of this variable is the zero
                    length OCTET STRING.

                    The private authentication key is NOT directly
                    represented by the value of this variable, but
                    rather it is represented according to an encoding.
                    This encoding is the bitwise exclusive-OR of the
                    old key with the new key, i.e., of the old private
                    authentication key (prior to the alteration) with
                    the new private authentication key (after the
                    alteration).  Thus, when processing a received
                    protocol Set operation, the new private
                    authentication key is obtained from the value of
                    this variable as the result of a bitwise
                    exclusive-OR of the variable's value and the old
                    private authentication key.  In calculating the
```

```
                exclusive-OR, if the old key is shorter than the
                new key, zero-valued padding is appended to the
                old key.  If no value for the old key exists, a
                zero-length OCTET STRING is used in the
                calculation."
            DEFVAL  { ''h }     -- the empty string
            ::= { partySecretsEntry 2 }

        partySecretsPrivPrivate OBJECT-TYPE
            SYNTAX  OCTET STRING   -- for desPrivProtocol: (SIZE (16))
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "An encoding of the party's private encryption key
                    which may be needed to support the privacy
                    protocol.  Although the value of this variable may
                    be altered by a management operation (e.g., a SNMP
                    Set-Request), its value can never be retrieved by
                    a management operation: when read, the value of
                    this variable is the zero length OCTET STRING.

                    The private encryption key is NOT directly
                    represented by the value of this variable, but
                    rather it is represented according to an encoding.
                    This encoding is the bitwise exclusive-OR of the
                    old key with the new key, i.e., of the old private
                    encryption key (prior to the alteration) with the
                    new private encryption key (after the alteration).
                    Thus, when processing a received protocol Set
                    operation, the new private encryption key is
                    obtained from the value of this variable as the
                    result of a bitwise exclusive-OR of the variable's
                    value and the old private encryption key.  In
                    calculating the exclusive-OR, if the old key is
                    shorter than the new key, zero-valued padding is
                    appended to the old key.  If no value for the old
                    key exists, a zero-length OCTET STRING is used in
                    the calculation."
            DEFVAL  { ''h }     -- the empty string
            ::= { partySecretsEntry 3 }

        partySecretsStatus OBJECT-TYPE
            SYNTAX  INTEGER  { valid(1), invalid(2) }
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "The status of the locally-held information on a
                    particular SNMP party.
```

```
                Setting an instance of this object to the value
                'valid(1)' has the effect of ensuring that valid
                local knowledge exists for the corresponding
                party.  For valid local knowledge to exist, there
                must be corresponding instances of each object in
                this table and in the partyTable.  Thus, the
                creation of instances in the partyTable (but not
                in the aclTable or viewTable) occurs as a direct
                result of the creation of instances in this table.

                Setting an instance of this object to the value
                'invalid(2)' has the effect of invalidating all
                local knowledge of the corresponding party,
                including the invalidating of any/all entries in
                the partyTable, the partySecretsTable, the
                aclTable, and the viewTable which reference said
                party.

                It is an implementation-specific matter as to
                whether the agent removes an invalidated entry
                from the table.  Accordingly, management stations
                must be prepared to receive from agents tabular
                information corresponding to entries not currently
                in use.  Proper interpretation of such entries
                requires examination of the relevant
                partySecretsStatus object."
        DEFVAL  { valid }
        ::= { partySecretsEntry 4 }


    --  The SNMP Access Privileges Database Group

    --  This group of objects allows the SNMP itself to be used to
    --  configure new SNMP parties, or to manipulate the access
    --  privileges of existing parties.
    --
    --  Implementation of the objects in this group is mandatory.


    aclTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF AclEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "The access privileges database."
    ::= { partyAccess 1 }
```

```
        aclEntry OBJECT-TYPE
            SYNTAX  AclEntry
            ACCESS  not-accessible
            STATUS  mandatory
            DESCRIPTION
                    "The access privileges for a particular requesting
                    SNMP party in accessing a particular target SNMP
                    party."
            INDEX  { aclTarget, aclSubject }
            ::= { aclTable 1 }

        AclEntry ::=
            SEQUENCE {
                aclTarget
                    Party,
                aclSubject
                    Party,
                aclPrivileges
                    INTEGER,
                aclStatus
                    INTEGER
            }

        aclTarget OBJECT-TYPE
            SYNTAX  Party
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "The target SNMP party whose performance of
                    management operations is constrained by this set
                    of access privileges."
            ::= { aclEntry 1 }

        aclSubject OBJECT-TYPE
            SYNTAX  Party
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "The subject SNMP party whose requests for
                    management operations to be performed is
                    constrained by this set of access privileges."
            ::= { aclEntry 2 }

        aclPrivileges OBJECT-TYPE
            SYNTAX  INTEGER (0..31)
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
```

```
                "The access privileges which govern what
                management operations a particular target party
                may perform when requested by a particular subject
                party.  These privileges are specified as a sum of
                values, where each value specifies a SNMP PDU type
                by which the subject party may request a permitted
                operation.  The value for a particular PDU type is
                computed as 2 raised to the value of the ASN.1
                context-specific tag for the appropriate SNMP PDU
                type.  The values (for the tags defined in RFC
                1157) are defined in RFC 1351 as:

                 Get         :   1
                 GetNext     :   2
                 GetResponse :   4
                 Set         :   8
                 Trap        :  16

                The null set is represented by the value zero."
            DEFVAL  { 3 }      -- Get & Get-Next
            ::= { aclEntry 3 }

        aclStatus OBJECT-TYPE
            SYNTAX  INTEGER  { valid(1), invalid(2) }
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "The status of the access privileges for a
                    particular requesting SNMP party in accessing a
                    particular target SNMP party.  Setting an instance
                    of this object to the value 'invalid(2)' has the
                    effect of invalidating the corresponding access
                    privileges.

                    It is an implementation-specific matter as to
                    whether the agent removes an invalidated entry
                    from the table.  Accordingly, management stations
                    must be prepared to receive from agents tabular
                    information corresponding to entries not currently
                    in use.  Proper interpretation of such entries
                    requires examination of the relevant aclStatus
                    object."
            DEFVAL  { valid }
            ::= { aclEntry 4 }
```

```
--   The MIB View Database Group

--  This group of objects allows the SNMP itself to be used to
--  configure new SNMP parties, or to manipulate the MIB
--  MIB views of existing parties.
--
--  Implementation of the objects in this group is mandatory.


viewTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF ViewEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "The table contained in the local database which
            defines local MIB views.  Each SNMP party has a
            single MIB view which is defined by two
            collections of view subtrees: the included view
            subtrees, and the excluded view subtrees.  Every
            such subtree, both included and excluded, is
            defined in this table.

            To determine if a particular object instance is in
            a particular SNMP party's MIB view, compare the
            object instance's Object Identifier with each
            entry (for this party) in this table.  If none
            match, then the object instance is not in the MIB
            view.  If one or more match, then the object
            instance is included in, or excluded from, the MIB
            view according to the value of viewStatus in the
            entry whose value of viewSubtree has the most
            sub-identifiers.  If multiple entries match and
            have the same number of sub-identifiers, then the
            lexicographically greatest instance of viewStatus
            determines the inclusion or exclusion.

            An object instance's Object Identifier X matches
            an entry in this table when the number of sub-
            identifiers in X is at least as many as in the
            value of viewSubtree for the entry, and each sub-
            identifier in the value of viewSubtree matches its
            corresponding sub-identifier in X.  Two sub-
            identifiers match either if the corresponding bit
            of viewMask is zero (the 'wild card' value), or if
            they are equal.

            Due to this 'wild card' capability, we introduce
            the term, a 'family' of view subtrees, to refer to
```

```
                the set of subtrees defined by a particular
                combination of values of viewSubtree and viewMask.
                In the case where no 'wild card' is defined in
                viewMask, the family of view subtrees reduces to a
                single view subtree."
        ::= { partyViews 1 }

        viewEntry OBJECT-TYPE
            SYNTAX  ViewEntry
            ACCESS  not-accessible
            STATUS  mandatory
            DESCRIPTION
                    "Information on a particular family of view
                    subtrees included in or excluded from a particular
                    SNMP party's MIB view."
            INDEX  { viewParty, viewSubtree }
            ::= { viewTable 1 }

        ViewEntry ::=
            SEQUENCE {
                viewParty
                    Party,
                viewSubtree
                    OBJECT IDENTIFIER,
                viewStatus
                    INTEGER,
                viewMask
                    OCTET STRING
            }

        viewParty  OBJECT-TYPE
            SYNTAX  Party
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "The SNMP party whose single MIB view includes or
                    excludes a particular family of view subtrees."
            ::= { viewEntry 1 }

        viewSubtree OBJECT-TYPE
            SYNTAX  OBJECT IDENTIFIER
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "The view subtree which, in combination with the
                    corresponding instance of viewMask, defines a
                    family of view subtrees.  This family is included
                    in, or excluded from the particular SNMP party's
```

```
                MIB view, according to the value of the
                corresponding instance of viewStatus."
        ::= { viewEntry 2 }

    viewStatus OBJECT-TYPE
        SYNTAX  INTEGER  {
                    included(1),
                    excluded(2),
                    invalid(3)
                }
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The status of a particular family of view
                subtrees within the particular SNMP party's MIB
                view.  The value 'included(1)' indicates that the
                corresponding instances of viewSubtree and
                viewMask define a family of view subtrees included
                in the MIB view.  The  value 'excluded(2)'
                indicates that the corresponding instances of
                viewSubtree and viewMask define a family of view
                subtrees excluded from the MIB view.

                Setting an instance of this object to the value
                'invalid(3)' has the effect of invalidating the
                presence or absence of the corresponding family of
                view subtrees in the corresponding SNMP party's
                MIB view.

                It is an implementation-specific matter as to
                whether the agent removes an invalidated entry
                from the table.  Accordingly, management stations
                must be prepared to receive from agents tabular
                information corresponding to entries not currently
                in use.  Proper interpretation of such entries
                requires examination of the relevant viewStatus
                object."
        DEFVAL  { included }
        ::= { viewEntry 3 }

    viewMask  OBJECT-TYPE
        SYNTAX  OCTET STRING (SIZE (0..16))
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "The bit mask which, in combination with the
                corresponding instance of viewSubtree, defines a
                family of view subtrees.
```

```
                    Each bit of this bit mask corresponds to a sub-
                    identifier of viewSubtree, with the most
                    significant bit of the i-th octet of this octet
                    string value (extended if necessary, see below)
                    corresponding to the (8*i - 7)-th sub-identifier,
                    and the least significant bit of the i-th octet of
                    this octet string corresponding to the (8*i)-th
                    sub-identifier, where i is in the range 1 through
                    16.

                    Each bit of this bit mask specifies whether or not
                    the corresponding sub-identifiers must match when
                    determining if an Object Identifier is in this
                    family of view subtrees; a '1' indicates that an
                    exact match must occur; a '0' indicates 'wild
                    card', i.e., any sub-identifier value matches.

                    Thus, the Object Identifier X of an object
                    instance is contained in a family of view subtrees
                    if the following criteria are met:

                         for each sub-identifier of the value of
                         viewSubtree, either:

                              the i-th bit of viewMask is 0, or

                              the i-th sub-identifier of X is equal to
                              the i-th sub-identifier of the value of
                              viewSubtree.

                    If the value of this bit mask is M bits long and
                    there are more than M sub-identifiers in the
                    corresponding instance of viewSubtree, then the
                    bit mask is extended with 1's to be the required
                    length.

                    Note that when the value of this object is the
                    zero-length string, this extension rule results in
                    a mask of all-1's being used (i.e., no 'wild
                    card'), and the family of view subtrees is the one
                    view subtree uniquely identified by the
                    corresponding instance of viewSubtree."
               DEFVAL  { ''h }
               ::= { viewEntry 4 }


          END
```

5. Acknowledgments

This document was produced on behalf of the SNMP Security Working Group of the Internet Engineering Task Force. The authors wish to thank the members of the working group, and others who contributed to this effort.

6. References

[1] Rose, M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP based internets", RFC 1155, Performance Systems International, Hughes LAN Systems, May 1990.

[2] McCloghrie, K., and M. Rose, "Management Information Base for Network Management of TCP/IP-based Internets", RFC 1156, Hughes LAN Systems and Performance Systems International, May 1990.

[3] Case, J., M. Fedor, M. Schoffstall, and J. Davin, The Simple Network Management Protocol", RFC 1157, University of Tennessee at Knoxville, Performance Systems International, Performance Systems International, and the MIT Laboratory for Computer Science, May 1990.

[4] McCloghrie K., and M. Rose, Editors, "Management Information Base for Network Management of TCP/IP-based internets", RFC 1213, Performance Systems International, March 1991.

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

[6] Information processing systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Notation One (ASN.1), International Organization for Standardization, International Standard 8825, December 1987.

[7] Rose, M., and K. McCloghrie, Editors, "Concise MIB Definitions", RFC 1212, Performance Systems International, Hughes LAN Systems, March 1991.

[8] Davin, J., Galvin, J., and K. McCloghrie, "SNMP Administrative Model", RFC 1351, MIT Laboratory for Computer Science, Trusted Information Systems, Inc., Hughes LAN Systems, Inc., July 1992.

[9] Galvin, J., McCloghrie, K., and J. Davin, "SNMP Security Protocols", RFC 1352, Trusted Information Systems, Inc., Hughes LAN Systems, Inc., MIT Laboratory for Computer Science, July

1992.

Security Considerstions

Security issues are discussed in section 3.1. and in RFCs 1351 and 1352.

Authors' Addresses

Keith McCloghrie
Hughes LAN Systems, Inc.
Mountain View, CA 94043

Phone: (415) 966-7934
EMail: kzm@hls.com

James R. Davin
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139

Phone: (617) 253-6020
EMail: jrd@ptt.lcs.mit.edu

James M. Galvin
Trusted Information Systems, Inc.
3060 Washington Road, Route 97
Glenwood, MD 21738

Phone: (301) 854-6889
EMail: galvin@tis.com