

Network Working Group
Request for Comments: 3415
STD: 62
Obsoletes: 2575
Category: Standards Track

B. Wijnen
Lucent Technologies
R. Presuhn
BMC Software, Inc.
K. McCloghrie
Cisco Systems, Inc.
December 2002

View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2002). All Rights Reserved.

Abstract

This document describes the View-based Access Control Model (VACM) for use in the Simple Network Management Protocol (SNMP) architecture. It defines the Elements of Procedure for controlling access to management information. This document also includes a Management Information Base (MIB) for remotely managing the configuration parameters for the View-based Access Control Model. This document obsoletes RFC 2575.

Table of Contents

1. Introduction	2
1.2. Access Control	3
1.3. Local Configuration Datastore	3
2. Elements of the Model	4
2.1. Groups	4
2.2. securityLevel	4
2.3. Contexts	4
2.4. MIB Views and View Families	5
2.4.1. View Subtree	5
2.4.2. ViewTreeFamily	6
2.5. Access Policy	6
3. Elements of Procedure	7
3.1. Overview of isAccessAllowed Process	8
3.2. Processing the isAccessAllowed Service Request	9
4. Definitions	11
5. Intellectual Property	28
6. Acknowledgements	28
7. Security Considerations	30
7.1. Recommended Practices	30
7.2. Defining Groups	30
7.3. Conformance	31
7.4. Access to the SNMP-VIEW-BASED-ACM-MIB	31
8. References	31
A. Installation	33
B. Change Log	36
Editors' Addresses	38
Full Copyright Statement	39

1. Introduction

The Architecture for describing Internet Management Frameworks [RFC3411] describes that an SNMP engine is composed of:

- 1) a Dispatcher
- 2) a Message Processing Subsystem,
- 3) a Security Subsystem, and
- 4) an Access Control Subsystem.

Applications make use of the services of these subsystems.

It is important to understand the SNMP architecture and its terminology to understand where the View-based Access Control Model described in this document fits into the architecture and interacts with other subsystems within the architecture. The reader is expected to have read and understood the description and terminology of the SNMP architecture, as defined in [RFC3411].

The Access Control Subsystem of an SNMP engine has the responsibility for checking whether a specific type of access (read, write, notify) to a particular object (instance) is allowed.

It is the purpose of this document to define a specific model of the Access Control Subsystem, designated the View-based Access Control Model. Note that this is not necessarily the only Access Control Model.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119.

1.2. Access Control

Access Control occurs (either implicitly or explicitly) in an SNMP entity when processing SNMP retrieval or modification request messages from an SNMP entity. For example a Command Responder application applies Access Control when processing requests that it received from a Command Generator application. These requests contain Read Class and Write Class PDUs as defined in [RFC3411].

Access Control also occurs in an SNMP entity when an SNMP notification message is generated (by a Notification Originator application). These notification messages contain Notification Class PDUs as defined in [RFC3411].

The View-based Access Control Model defines a set of services that an application (such as a Command Responder or a Notification Originator application) can use for checking access rights. It is the responsibility of the application to make the proper service calls for access checking.

1.3. Local Configuration Datastore

To implement the model described in this document, an SNMP entity needs to retain information about access rights and policies. This information is part of the SNMP engine's Local Configuration Datastore (LCD). See [RFC3411] for the definition of LCD.

In order to allow an SNMP entity's LCD to be remotely configured, portions of the LCD need to be accessible as managed objects. A MIB module, the View-based Access Control Model Configuration MIB, which defines these managed object types is included in this document.

2. Elements of the Model

This section contains definitions to realize the access control service provided by the View-based Access Control Model.

2.1. Groups

A group is a set of zero or more <securityModel, securityName> tuples on whose behalf SNMP management objects can be accessed. A group defines the access rights afforded to all securityNames which belong to that group. The combination of a securityModel and a securityName maps to at most one group. A group is identified by a groupName.

The Access Control module assumes that the securityName has already been authenticated as needed and provides no further authentication of its own.

The View-based Access Control Model uses the securityModel and the securityName as inputs to the Access Control module when called to check for access rights. It determines the groupName as a function of securityModel and securityName.

2.2. securityLevel

Different access rights for members of a group can be defined for different levels of security, i.e., noAuthNoPriv, authNoPriv, and authPriv. The securityLevel identifies the level of security that will be assumed when checking for access rights. See the SNMP Architecture document [RFC3411] for a definition of securityLevel.

The View-based Access Control Model requires that the securityLevel is passed as input to the Access Control module when called to check for access rights.

2.3. Contexts

An SNMP context is a collection of management information accessible by an SNMP entity. An item of management information may exist in more than one context. An SNMP entity potentially has access to many contexts. Details about the naming of management information can be found in the SNMP Architecture document [RFC3411].

The View-based Access Control Model defines a vacmContextTable that lists the locally available contexts by contextName.

2.4. MIB Views and View Families

For security reasons, it is often valuable to be able to restrict the access rights of some groups to only a subset of the management information in the management domain. To provide this capability, access to a context is via a "MIB view" which details a specific set of managed object types (and optionally, the specific instances of object types) within that context. For example, for a given context, there will typically always be one MIB view which provides access to all management information in that context, and often there will be other MIB views each of which contains some subset of the information. So, the access allowed for a group can be restricted in the desired manner by specifying its rights in terms of the particular (subset) MIB view it can access within each appropriate context.

Since managed object types (and their instances) are identified via the tree-like naming structure of ISO's OBJECT IDENTIFIERS [ISO-ASN.1, RFC2578], it is convenient to define a MIB view as the combination of a set of "view subtrees", where each view subtree is a subtree within the managed object naming tree. Thus, a simple MIB view (e.g., all managed objects within the Internet Network Management Framework) can be defined as a single view subtree, while more complicated MIB views (e.g., all information relevant to a particular network interface) can be represented by the union of multiple view subtrees.

While any set of managed objects can be described by the union of some number of view subtrees, situations can arise that would require a very large number of view subtrees. This could happen, for example, when specifying all columns in one conceptual row of a MIB table because they would appear in separate subtrees, one per column, each with a very similar format. Because the formats are similar, the required set of subtrees can easily be aggregated into one structure. This structure is named a family of view subtrees after the set of subtrees that it conceptually represents. A family of view subtrees can either be included or excluded from a MIB view.

2.4.1. View Subtree

A view subtree is the set of all MIB object instances which have a common ASN.1 OBJECT IDENTIFIER prefix to their names. A view subtree is identified by the OBJECT IDENTIFIER value which is the longest OBJECT IDENTIFIER prefix common to all (potential) MIB object instances in that subtree.

2.4.2. ViewTreeFamily

A family of view subtrees is a pairing of an OBJECT IDENTIFIER value (called the family name) together with a bit string value (called the family mask). The family mask indicates which sub-identifiers of the associated family name are significant to the family's definition.

For each possible managed object instance, that instance belongs to a particular ViewTreeFamily if both of the following conditions are true:

- the OBJECT IDENTIFIER name of the managed object instance contains at least as many sub-identifiers as does the family name, and
- each sub-identifier in the OBJECT IDENTIFIER name of the managed object instance matches the corresponding sub-identifier of the family name whenever the corresponding bit of the associated family mask is non-zero.

When the configured value of the family mask is all ones, the view subtree family is identical to the single view subtree identified by the family name.

When the configured value of the family mask is shorter than required to perform the above test, its value is implicitly extended with ones. Consequently, a view subtree family having a family mask of zero length always corresponds to a single view subtree.

2.5. Access Policy

The View-based Access Control Model determines the access rights of a group, representing zero or more securityNames which have the same access rights. For a particular context, identified by contextName, to which a group, identified by groupName, has access using a particular securityModel and securityLevel, that group's access rights are given by a read-view, a write-view and a notify-view.

The read-view represents the set of object instances authorized for the group when reading objects. Reading objects occurs when processing a retrieval operation (when handling Read Class PDUs).

The write-view represents the set of object instances authorized for the group when writing objects. Writing objects occurs when processing a write operation (when handling Write Class PDUs).

The notify-view represents the set of object instances authorized for the group when sending objects in a notification, such as when sending a notification (when sending Notification Class PDUs).

3. Elements of Procedure

This section describes the procedures followed by an Access Control module that implements the View-based Access Control Model when checking access rights as requested by an application (for example a Command Responder or a Notification Originator application). The abstract service primitive is:

```

statusInformation =          -- success or errorIndication
    isAccessAllowed(
        securityModel        -- Security Model in use
        securityName         -- principal who wants access
        securityLevel        -- Level of Security
        viewType             -- read, write, or notify view
        contextName          -- context containing variableName
        variableName         -- OID for the managed object
    )

```

The abstract data elements are:

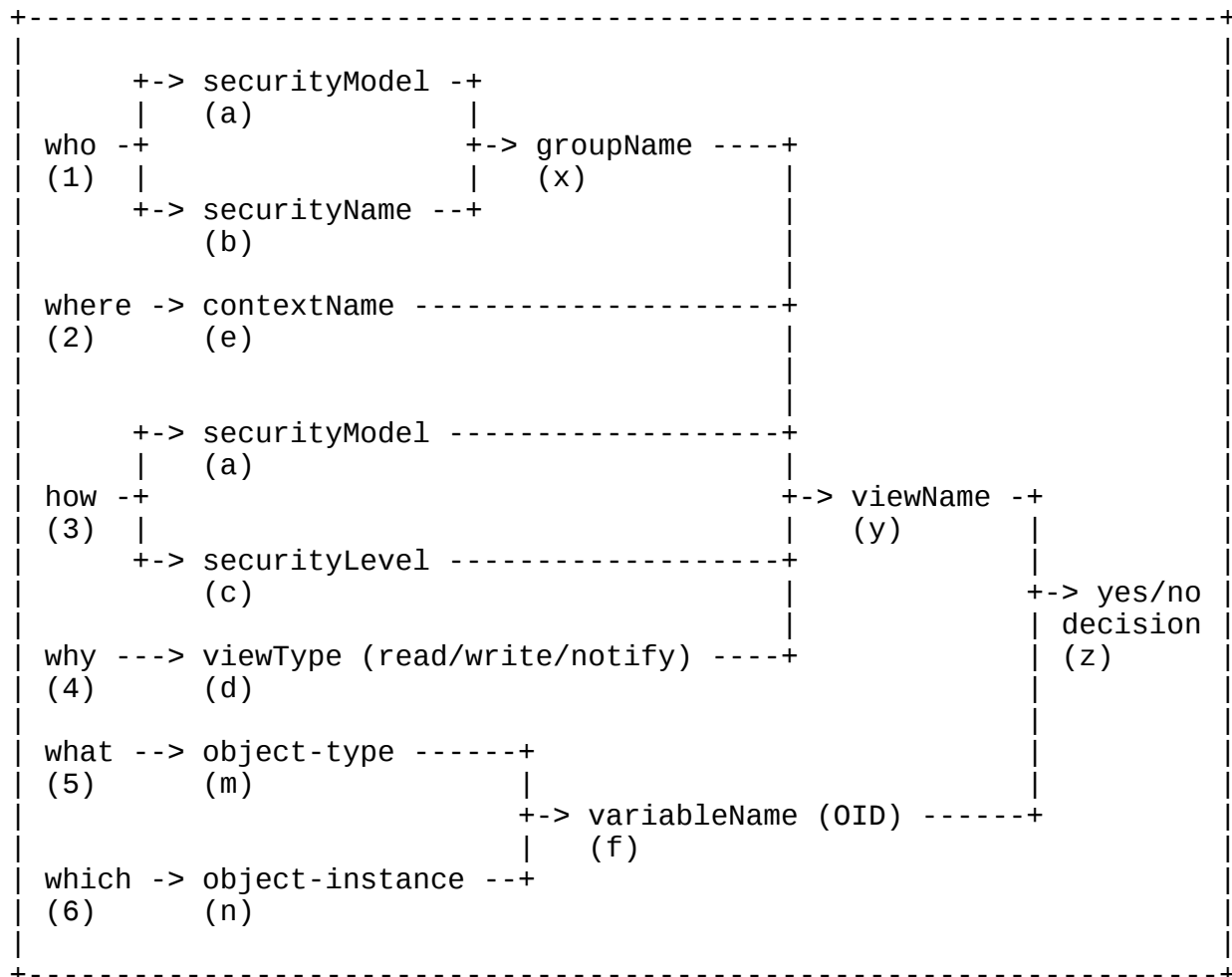
```

statusInformation - one of the following:
    accessAllowed - a MIB view was found and access is granted.
    notInView     - a MIB view was found but access is denied.
                   The variableName is not in the configured
                   MIB view for the specified viewType (e.g., in
                   the relevant entry in the vacmAccessTable).
    noSuchView    - no MIB view found because no view has been
                   configured for specified viewType (e.g., in
                   the relevant entry in the vacmAccessTable).
    noSuchContext - no MIB view found because of no entry in the
                   vacmContextTable for specified contextName.
    noGroupName   - no MIB view found because no entry has been
                   configured in the vacmSecurityToGroupTable
                   for the specified combination of
                   securityModel and securityName.
    noAccessEntry - no MIB view found because no entry has been
                   configured in the vacmAccessTable for the
                   specified combination of contextName,
                   groupName (from vacmSecurityToGroupTable),
                   securityModel and securityLevel.
    otherError    - failure, an undefined error occurred.
securityModel    - Security Model under which access is requested.
securityName     - the principal on whose behalf access is requested.
securityLevel    - Level of Security under which access is requested.
viewType         - view to be checked (read, write or notify).
contextName      - context in which access is requested.
variableName     - object instance to which access is requested.

```

3.1. Overview of isAccessAllowed Process

The following picture shows how the decision for access control is made by the View-based Access Control Model.



How the decision for `isAccessAllowed` is made.

1) Inputs to the `isAccessAllowed` service are:

- (a) `securityModel` -- Security Model in use
- (b) `securityName` -- principal who wants to access
- (c) `securityLevel` -- Level of Security
- (d) `viewType` -- read, write, or notify view
- (e) `contextName` -- context containing `variableName`
- (f) `variableName` -- OID for the managed object
 - this is made up of:
 - object-type (m)
 - object-instance (n)

2) The partial "who" (1), represented by the `securityModel` (a) and the `securityName` (b), are used as the indices (a,b) into the `vacmSecurityToGroupTable` to find a single entry that produces a group, represented by `groupName` (x).

3) The "where" (2), represented by the `contextName` (e), the "who", represented by the `groupName` (x) from the previous step, and the "how" (3), represented by `securityModel` (a) and `securityLevel` (c), are used as indices (e,x,a,c) into the `vacmAccessTable` to find a single entry that contains three MIB views.

4) The "why" (4), represented by the `viewType` (d), is used to select the proper MIB view, represented by a `viewName` (y), from the `vacmAccessEntry` selected in the previous step. This `viewName` (y) is an index into the `vacmViewTreeFamilyTable` and selects the set of entries that define the `variableNames` which are included in or excluded from the MIB view identified by the `viewName` (y).

5) The "what" (5) type of management data and "which" (6) particular instance, represented by the `variableName` (f), is then checked to be in the MIB view or not, e.g., the yes/no decision (z).

3.2. Processing the `isAccessAllowed` Service Request

This section describes the procedure followed by an Access Control module that implements the View-based Access Control Model whenever it receives an `isAccessAllowed` request.

1) The `vacmContextTable` is consulted for information about the SNMP context identified by the `contextName`. If information about this SNMP context is absent from the table, then an `errorIndication` (`noSuchContext`) is returned to the calling module.

- 2) The `vacmSecurityToGroupTable` is consulted for mapping the `securityModel` and `securityName` to a `groupName`. If the information about this combination is absent from the table, then an `errorIndication` (`noGroupName`) is returned to the calling module.
- 3) The `vacmAccessTable` is consulted for information about the `groupName`, `contextName`, `securityModel` and `securityLevel`. If information about this combination is absent from the table, then an `errorIndication` (`noAccessEntry`) is returned to the calling module.
- 4)
 - a) If the `viewType` is "read", then the read view is used for checking access rights.
 - b) If the `viewType` is "write", then the write view is used for checking access rights.
 - c) If the `viewType` is "notify", then the notify view is used for checking access rights.

If the view to be used is the empty view (zero length `viewName`) then an `errorIndication` (`noSuchView`) is returned to the calling module.

- 5)
 - a) If there is no view configured for the specified `viewType`, then an `errorIndication` (`noSuchView`) is returned to the calling module.
 - b) If the specified `variableName` (object instance) is not in the MIB view (see DESCRIPTION clause for `vacmViewTreeFamilyTable` in section 4), then an `errorIndication` (`notInView`) is returned to the calling module.

Otherwise,

- c) The specified `variableName` is in the MIB view. A status information of success (`accessAllowed`) is returned to the calling module.

4. Definitions

SNMP-VIEW-BASED-ACM-MIB DEFINITIONS ::= BEGIN

IMPORTS

```

MODULE-COMPLIANCE, OBJECT-GROUP          FROM SNMPv2-CONF
MODULE-IDENTITY, OBJECT-TYPE,
snmpModules                               FROM SNMPv2-SMI
TestAndIncr,
RowStatus, StorageType                   FROM SNMPv2-TC
SnmpAdminString,
SnmpSecurityLevel,
SnmpSecurityModel                         FROM SNMP-FRAMEWORK-MIB;
```

```

snmpVacmMIB      MODULE-IDENTITY
LAST-UPDATED "200210160000Z"          -- 16 Oct 2002, midnight
ORGANIZATION "SNMPv3 Working Group"
CONTACT-INFO "WG-email:  snmpv3@lists.tislabs.com
               Subscribe:  majordomo@lists.tislabs.com
               In message body:  subscribe snmpv3

               Co-Chair:  Russ Mundy
                           Network Associates Laboratories
postal:  15204 Omega Drive, Suite 300
          Rockville, MD 20850-4601
          USA
email:   mundy@tislabs.com
phone:   +1 301-947-7107

               Co-Chair:  David Harrington
                           Enterasys Networks
Postal:   35 Industrial Way
          P. O. Box 5004
          Rochester, New Hampshire 03866-5005
          USA
EMail:   dbh@enterasys.com
Phone:   +1 603-337-2614

               Co-editor:  Bert Wijnen
                           Lucent Technologies
postal:   Schagen 33
          3461 GL Linschoten
          Netherlands
email:    bwijnen@lucent.com
phone:    +31-348-480-685

               Co-editor:  Randy Presuhn
                           BMC Software, Inc.
```

postal: 2141 North First Street
 San Jose, CA 95131
 USA
 email: randy_presuhn@bmc.com
 phone: +1 408-546-1006

Co-editor: Keith McCloghrie
 Cisco Systems, Inc.
 postal: 170 West Tasman Drive
 San Jose, CA 95134-1706
 USA
 email: kzm@cisco.com
 phone: +1-408-526-5260

"

DESCRIPTION "The management information definitions for the
 View-based Access Control Model for SNMP.

Copyright (C) The Internet Society (2002). This
 version of this MIB module is part of RFC 3415;
 see the RFC itself for full legal notices.

"

-- Revision history

REVISION "200210160000Z" -- 16 Oct 2002, midnight
 DESCRIPTION "Clarifications, published as RFC3415"

REVISION "199901200000Z" -- 20 Jan 1999, midnight
 DESCRIPTION "Clarifications, published as RFC2575"

REVISION "199711200000Z" -- 20 Nov 1997, midnight
 DESCRIPTION "Initial version, published as RFC2275"

::= { snmpModules 16 }

-- Administrative assignments *****

vacmMIBObjects OBJECT IDENTIFIER ::= { snmpVacmMIB 1 }
 vacmMIBConformance OBJECT IDENTIFIER ::= { snmpVacmMIB 2 }

-- Information about Local Contexts *****

vacmContextTable OBJECT-TYPE
 SYNTAX SEQUENCE OF VacmContextEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION "The table of locally available contexts.

This table provides information to SNMP Command

Generator applications so that they can properly configure the vacmAccessTable to control access to all contexts at the SNMP entity.

This table may change dynamically if the SNMP entity allows that contexts are added/deleted dynamically (for instance when its configuration changes). Such changes would happen only if the management instrumentation at that SNMP entity recognizes more (or fewer) contexts.

The presence of entries in this table and of entries in the vacmAccessTable are independent. That is, a context identified by an entry in this table is not necessarily referenced by any entries in the vacmAccessTable; and the context(s) referenced by an entry in the vacmAccessTable does not necessarily currently exist and thus need not be identified by an entry in this table.

This table must be made accessible via the default context so that Command Responder applications have a standard way of retrieving the information.

This table is read-only. It cannot be configured via SNMP.

"

::= { vacmMIBObjects 1 }

```
vacmContextEntry OBJECT-TYPE
    SYNTAX      VacmContextEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION  "Information about a particular context."
    INDEX        {
        vacmContextName
    }
    ::= { vacmContextTable 1 }
```

```
VacmContextEntry ::= SEQUENCE
{
    vacmContextName SnmpAdminString
}
```

```
vacmContextName OBJECT-TYPE
    SYNTAX      SnmpAdminString (SIZE(0..32))
    MAX-ACCESS   read-only
    STATUS       current
```

DESCRIPTION "A human readable name identifying a particular context at a particular SNMP entity.

The empty contextName (zero length) represents the default context.

"

::= { vacmContextEntry 1 }

-- Information about Groups *****

vacmSecurityToGroupTable OBJECT-TYPE

SYNTAX SEQUENCE OF VacmSecurityToGroupEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION "This table maps a combination of securityModel and securityName into a groupName which is used to define an access control policy for a group of principals.

"

::= { vacmMIBObjects 2 }

vacmSecurityToGroupEntry OBJECT-TYPE

SYNTAX VacmSecurityToGroupEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION "An entry in this table maps the combination of a securityModel and securityName into a groupName.

"

INDEX {
 vacmSecurityModel,
 vacmSecurityName
}

::= { vacmSecurityToGroupTable 1 }

VacmSecurityToGroupEntry ::= SEQUENCE

```
{
    vacmSecurityModel          SnmpSecurityModel,
    vacmSecurityName           SnmpAdminString,
    vacmGroupName              SnmpAdminString,
    vacmSecurityToGroupStorageType StorageType,
    vacmSecurityToGroupStatus   RowStatus
}
```

vacmSecurityModel OBJECT-TYPE

SYNTAX SnmpSecurityModel(1..2147483647)

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION "The Security Model, by which the vacmSecurityName referenced by this entry is provided.

Note, this object may not take the 'any' (0) value.
"

::= { vacmSecurityToGroupEntry 1 }

vacmSecurityName OBJECT-TYPE

SYNTAX SnmpAdminString (SIZE(1..32))

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION "The securityName for the principal, represented in a Security Model independent format, which is mapped by this entry to a groupName.
"

::= { vacmSecurityToGroupEntry 2 }

vacmGroupName OBJECT-TYPE

SYNTAX SnmpAdminString (SIZE(1..32))

MAX-ACCESS read-create

STATUS current

DESCRIPTION "The name of the group to which this entry (e.g., the combination of securityModel and securityName) belongs.

This groupName is used as index into the vacmAccessTable to select an access control policy. However, a value in this table does not imply that an instance with the value exists in table vacmAccessTable.
"

::= { vacmSecurityToGroupEntry 3 }

vacmSecurityToGroupStorageType OBJECT-TYPE

SYNTAX StorageType

MAX-ACCESS read-create

STATUS current

DESCRIPTION "The storage type for this conceptual row. Conceptual rows having the value 'permanent' need not allow write-access to any columnar objects in the row.
"

DEFVAL { nonVolatile }

::= { vacmSecurityToGroupEntry 4 }

vacmSecurityToGroupStatus OBJECT-TYPE

SYNTAX RowStatus

MAX-ACCESS read-create

STATUS current

DESCRIPTION "The status of this conceptual row.

Until instances of all corresponding columns are appropriately configured, the value of the

corresponding instance of the vacmSecurityToGroupStatus column is 'notReady'.

In particular, a newly created row cannot be made active until a value has been set for vacmGroupName.

The RowStatus TC [RFC2579] requires that this DESCRIPTION clause states under which circumstances other objects in this row can be modified:

The value of this object has no effect on whether other objects in this conceptual row can be modified.
"

::= { vacmSecurityToGroupEntry 5 }

-- Information about Access Rights *****

```
vacmAccessTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VacmAccessEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION  "The table of access rights for groups.
```

Each entry is indexed by a groupName, a contextPrefix, a securityModel and a securityLevel. To determine whether access is allowed, one entry from this table needs to be selected and the proper viewName from that entry must be used for access control checking.

To select the proper entry, follow these steps:

- 1) the set of possible matches is formed by the intersection of the following sets of entries:

```
the set of entries with identical vacmGroupName
the union of these two sets:
- the set with identical vacmAccessContextPrefix
- the set of entries with vacmAccessContextMatch
  value of 'prefix' and matching
  vacmAccessContextPrefix
intersected with the union of these two sets:
- the set of entries with identical
  vacmSecurityModel
- the set of entries with vacmSecurityModel
  value of 'any'
intersected with the set of entries with
vacmAccessSecurityLevel value less than or equal
to the requested securityLevel
```

- 2) if this set has only one member, we're done
otherwise, it comes down to deciding how to weight
the preferences between ContextPrefixes,
SecurityModels, and SecurityLevels as follows:
 - a) if the subset of entries with securityModel
matching the securityModel in the message is
not empty, then discard the rest.
 - b) if the subset of entries with
vacmAccessContextPrefix matching the contextName
in the message is not empty,
then discard the rest
 - c) discard all entries with ContextPrefixes shorter
than the longest one remaining in the set
 - d) select the entry with the highest securityLevel

Please note that for securityLevel noAuthNoPriv, all
groups are really equivalent since the assumption that
the securityName has been authenticated does not hold.

"

::= { vacmMIBObjects 4 }

vacmAccessEntry OBJECT-TYPE
SYNTAX VacmAccessEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION "An access right configured in the Local Configuration
Datastore (LCD) authorizing access to an SNMP context.

Entries in this table can use an instance value for
object vacmGroupName even if no entry in table
vacmAccessSecurityToGroupTable has a corresponding
value for object vacmGroupName.

"

INDEX { vacmGroupName,
vacmAccessContextPrefix,
vacmAccessSecurityModel,
vacmAccessSecurityLevel
}

::= { vacmAccessTable 1 }

VacmAccessEntry ::= SEQUENCE
{
vacmAccessContextPrefix SnmpAdminString,
vacmAccessSecurityModel SnmpSecurityModel,
vacmAccessSecurityLevel SnmpSecurityLevel,
vacmAccessContextMatch INTEGER,
vacmAccessReadViewName SnmpAdminString,
vacmAccessWriteViewName SnmpAdminString,

```

        vacmAccessNotifyViewName    SnmpAdminString,
        vacmAccessStorageType       StorageType,
        vacmAccessStatus             RowStatus
    }

```

vacmAccessContextPrefix OBJECT-TYPE

```

SYNTAX      SnmpAdminString (SIZE(0..32))
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION "In order to gain the access rights allowed by this
            conceptual row, a contextName must match exactly
            (if the value of vacmAccessContextMatch is 'exact')
            or partially (if the value of vacmAccessContextMatch
            is 'prefix') to the value of the instance of this
            object.
            "
 ::= { vacmAccessEntry 1 }

```

vacmAccessSecurityModel OBJECT-TYPE

```

SYNTAX      SnmpSecurityModel
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION "In order to gain the access rights allowed by this
            conceptual row, this securityModel must be in use.
            "
 ::= { vacmAccessEntry 2 }

```

vacmAccessSecurityLevel OBJECT-TYPE

```

SYNTAX      SnmpSecurityLevel
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION "The minimum level of security required in order to
            gain the access rights allowed by this conceptual
            row. A securityLevel of noAuthNoPriv is less than
            authNoPriv which in turn is less than authPriv.

            If multiple entries are equally indexed except for
            this vacmAccessSecurityLevel index, then the entry
            which has the highest value for
            vacmAccessSecurityLevel is selected.
            "
 ::= { vacmAccessEntry 3 }

```

vacmAccessContextMatch OBJECT-TYPE

```

SYNTAX      INTEGER
            { exact (1), -- exact match of prefix and contextName
              prefix (2) -- Only match to the prefix
            }

```

```

MAX-ACCESS    read-create
STATUS        current
DESCRIPTION   "If the value of this object is exact(1), then all
               rows where the contextName exactly matches
               vacmAccessContextPrefix are selected.

               If the value of this object is prefix(2), then all
               rows where the contextName whose starting octets
               exactly match vacmAccessContextPrefix are selected.
               This allows for a simple form of wildcarding.
               "
DEFVAL        { exact }
::= { vacmAccessEntry 4 }

```

vacmAccessReadViewName OBJECT-TYPE

```

SYNTAX        SnmpAdminString (SIZE(0..32))
MAX-ACCESS    read-create
STATUS        current
DESCRIPTION   "The value of an instance of this object identifies
               the MIB view of the SNMP context to which this
               conceptual row authorizes read access.

               The identified MIB view is that one for which the
               vacmViewTreeFamilyViewName has the same value as the
               instance of this object; if the value is the empty
               string or if there is no active MIB view having this
               value of vacmViewTreeFamilyViewName, then no access
               is granted.
               "
DEFVAL        { ''H }    -- the empty string
::= { vacmAccessEntry 5 }

```

vacmAccessWriteViewName OBJECT-TYPE

```

SYNTAX        SnmpAdminString (SIZE(0..32))
MAX-ACCESS    read-create
STATUS        current
DESCRIPTION   "The value of an instance of this object identifies
               the MIB view of the SNMP context to which this
               conceptual row authorizes write access.

               The identified MIB view is that one for which the
               vacmViewTreeFamilyViewName has the same value as the
               instance of this object; if the value is the empty
               string or if there is no active MIB view having this
               value of vacmViewTreeFamilyViewName, then no access
               is granted.
               "
DEFVAL        { ''H }    -- the empty string

```

```
::= { vacmAccessEntry 6 }
```

```
vacmAccessNotifyViewName OBJECT-TYPE
```

```
SYNTAX      SmpAdminString (SIZE(0..32))
```

```
MAX-ACCESS  read-create
```

```
STATUS      current
```

```
DESCRIPTION "The value of an instance of this object identifies
              the MIB view of the SNMP context to which this
              conceptual row authorizes access for notifications.
```

```
              The identified MIB view is that one for which the
              vacmViewTreeFamilyViewName has the same value as the
              instance of this object; if the value is the empty
              string or if there is no active MIB view having this
              value of vacmViewTreeFamilyViewName, then no access
              is granted.
```

```
"
```

```
DEFVAL      { ''H }  -- the empty string
```

```
::= { vacmAccessEntry 7 }
```

```
vacmAccessStorageType OBJECT-TYPE
```

```
SYNTAX      StorageType
```

```
MAX-ACCESS  read-create
```

```
STATUS      current
```

```
DESCRIPTION "The storage type for this conceptual row.
```

```
              Conceptual rows having the value 'permanent' need not
              allow write-access to any columnar objects in the row.
```

```
"
```

```
DEFVAL      { nonVolatile }
```

```
::= { vacmAccessEntry 8 }
```

```
vacmAccessStatus OBJECT-TYPE
```

```
SYNTAX      RowStatus
```

```
MAX-ACCESS  read-create
```

```
STATUS      current
```

```
DESCRIPTION "The status of this conceptual row.
```

```
              The RowStatus TC [RFC2579] requires that this
              DESCRIPTION clause states under which circumstances
              other objects in this row can be modified:
```

```
              The value of this object has no effect on whether
              other objects in this conceptual row can be modified.
```

```
"
```

```
::= { vacmAccessEntry 9 }
```

```
-- Information about MIB views *****
```

```
-- Support for instance-level granularity is optional.
--
-- In some implementations, instance-level access control
-- granularity may come at a high performance cost. Managers
-- should avoid requesting such configurations unnecessarily.
```

```
vacmMIBViews      OBJECT IDENTIFIER ::= { vacmMIBObjects 5 }
```

```
vacmViewSpinLock  OBJECT-TYPE
```

```
    SYNTAX          TestAndIncr
```

```
    MAX-ACCESS      read-write
```

```
    STATUS          current
```

```
    DESCRIPTION     "An advisory lock used to allow cooperating SNMP
    Command Generator applications to coordinate their
    use of the Set operation in creating or modifying
    views.
```

When creating a new view or altering an existing view, it is important to understand the potential interactions with other uses of the view. The vacmViewSpinLock should be retrieved. The name of the view to be created should be determined to be unique by the SNMP Command Generator application by consulting the vacmViewTreeFamilyTable. Finally, the named view may be created (Set), including the advisory lock.

If another SNMP Command Generator application has altered the views in the meantime, then the spin lock's value will have changed, and so this creation will fail because it will specify the wrong value for the spin lock.

Since this is an advisory lock, the use of this lock is not enforced.

"

```
::= { vacmMIBViews 1 }
```

```
vacmViewTreeFamilyTable OBJECT-TYPE
```

```
    SYNTAX          SEQUENCE OF VacmViewTreeFamilyEntry
```

```
    MAX-ACCESS      not-accessible
```

```
    STATUS          current
```

```
    DESCRIPTION     "Locally held information about families of subtrees
    within MIB views.
```

Each MIB view is defined by two sets of view subtrees:

- the included view subtrees, and
- the excluded view subtrees.

Every such view subtree, both the included and the

excluded ones, is defined in this table.

To determine if a particular object instance is in a particular MIB view, compare the object instance's OBJECT IDENTIFIER with each of the MIB view's active entries 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 vacmViewTreeFamilyType in the entry whose value of vacmViewTreeFamilySubtree has the most sub-identifiers. If multiple entries match and have the same number of sub-identifiers (when wildcarding is specified with the value of vacmViewTreeFamilyMask), then the lexicographically greatest instance of vacmViewTreeFamilyType determines the inclusion or exclusion.

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

A 'family' of subtrees is the set of subtrees defined by a particular combination of values of vacmViewTreeFamilySubtree and vacmViewTreeFamilyMask.

In the case where no 'wild card' is defined in the vacmViewTreeFamilyMask, the family of subtrees reduces to a single subtree.

When creating or changing MIB views, an SNMP Command Generator application should utilize the vacmViewSpinLock to try to avoid collisions. See DESCRIPTION clause of vacmViewSpinLock.

When creating MIB views, it is strongly advised that first the 'excluded' vacmViewTreeFamilyEntries are created and then the 'included' entries.

When deleting MIB views, it is strongly advised that first the 'included' vacmViewTreeFamilyEntries are

deleted and then the 'excluded' entries.

If a create for an entry for instance-level access control is received and the implementation does not support instance-level granularity, then an inconsistentName error must be returned.

"

::= { vacmMIBViews 2 }

vacmViewTreeFamilyEntry OBJECT-TYPE

SYNTAX VacmViewTreeFamilyEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION "Information on a particular family of view subtrees included in or excluded from a particular SNMP context's MIB view.

Implementations must not restrict the number of families of view subtrees for a given MIB view, except as dictated by resource constraints on the overall number of entries in the vacmViewTreeFamilyTable.

If no conceptual rows exist in this table for a given MIB view (viewName), that view may be thought of as consisting of the empty set of view subtrees.

"

INDEX { vacmViewTreeFamilyViewName,
vacmViewTreeFamilySubtree
}

::= { vacmViewTreeFamilyTable 1 }

VacmViewTreeFamilyEntry ::= SEQUENCE

```
{
    vacmViewTreeFamilyViewName      SnmpAdminString,
    vacmViewTreeFamilySubtree       OBJECT IDENTIFIER,
    vacmViewTreeFamilyMask          OCTET STRING,
    vacmViewTreeFamilyType          INTEGER,
    vacmViewTreeFamilyStorageType   StorageType,
    vacmViewTreeFamilyStatus        RowStatus
}
```

vacmViewTreeFamilyViewName OBJECT-TYPE

SYNTAX SnmpAdminString (SIZE(1..32))

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION "The human readable name for a family of view subtrees.

"

```
::= { vacmViewTreeFamilyEntry 1 }
```

```
vacmViewTreeFamilySubtree OBJECT-TYPE
```

```
SYNTAX          OBJECT IDENTIFIER
```

```
MAX-ACCESS      not-accessible
```

```
STATUS          current
```

```
DESCRIPTION     "The MIB subtree which when combined with the  
corresponding instance of vacmViewTreeFamilyMask  
defines a family of view subtrees.  
"
```

```
::= { vacmViewTreeFamilyEntry 2 }
```

```
vacmViewTreeFamilyMask OBJECT-TYPE
```

```
SYNTAX          OCTET STRING (SIZE (0..16))
```

```
MAX-ACCESS      read-create
```

```
STATUS          current
```

```
DESCRIPTION     "The bit mask which, in combination with the  
corresponding instance of vacmViewTreeFamilySubtree,  
defines a family of view subtrees.
```

Each bit of this bit mask corresponds to a sub-identifier of vacmViewTreeFamilySubtree, 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, for each sub-identifier of the value of vacmViewTreeFamilySubtree, either:

the i-th bit of vacmViewTreeFamilyMask is 0, or

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

If the value of this bit mask is M bits long and

there are more than M sub-identifiers in the corresponding instance of vacmViewTreeFamilySubtree, 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 vacmViewTreeFamilySubtree.

Note that masks of length greater than zero length do not need to be supported. In this case this object is made read-only.

"

```
DEFVAL      { ''H }
 ::= { vacmViewTreeFamilyEntry 3 }
```

vacmViewTreeFamilyType OBJECT-TYPE

```
SYNTAX      INTEGER { included(1), excluded(2) }
MAX-ACCESS   read-create
STATUS       current
DESCRIPTION  "Indicates whether the corresponding instances of
               vacmViewTreeFamilySubtree and vacmViewTreeFamilyMask
               define a family of view subtrees which is included in
               or excluded from the MIB view.
```

"

```
DEFVAL      { included }
 ::= { vacmViewTreeFamilyEntry 4 }
```

vacmViewTreeFamilyStorageType OBJECT-TYPE

```
SYNTAX      StorageType
MAX-ACCESS   read-create
STATUS       current
DESCRIPTION  "The storage type for this conceptual row.
```

Conceptual rows having the value 'permanent' need not allow write-access to any columnar objects in the row.

"

```
DEFVAL      { nonVolatile }
 ::= { vacmViewTreeFamilyEntry 5 }
```

vacmViewTreeFamilyStatus OBJECT-TYPE

```
SYNTAX      RowStatus
MAX-ACCESS   read-create
STATUS       current
DESCRIPTION  "The status of this conceptual row.
```

The RowStatus TC [RFC2579] requires that this DESCRIPTION clause states under which circumstances other objects in this row can be modified:

The value of this object has no effect on whether other objects in this conceptual row can be modified.
"

::= { vacmViewTreeFamilyEntry 6 }

-- Conformance information *****

vacmMIBCompliances OBJECT IDENTIFIER ::= { vacmMIBConformance 1 }
vacmMIBGroups OBJECT IDENTIFIER ::= { vacmMIBConformance 2 }

-- Compliance statements *****

vacmMIBCompliance MODULE-COMPLIANCE

STATUS current

DESCRIPTION "The compliance statement for SNMP engines which implement the SNMP View-based Access Control Model configuration MIB.
"

MODULE -- this module

MANDATORY-GROUPS { vacmBasicGroup }

OBJECT vacmAccessContextMatch
MIN-ACCESS read-only
DESCRIPTION "Write access is not required."

OBJECT vacmAccessReadViewName
MIN-ACCESS read-only
DESCRIPTION "Write access is not required."

OBJECT vacmAccessWriteViewName
MIN-ACCESS read-only
DESCRIPTION "Write access is not required."

OBJECT vacmAccessNotifyViewName
MIN-ACCESS read-only
DESCRIPTION "Write access is not required."

OBJECT vacmAccessStorageType
MIN-ACCESS read-only
DESCRIPTION "Write access is not required."

OBJECT vacmAccessStatus
MIN-ACCESS read-only
DESCRIPTION "Create/delete/modify access to the

```
vacmAccessTable is not required.
"
```

```
OBJECT      vacmViewTreeFamilyMask
WRITE-SYNTAX OCTET STRING (SIZE (0))
MIN-ACCESS   read-only
DESCRIPTION  "Support for configuration via SNMP of subtree
              families using wild-cards is not required.
"
```

```
OBJECT      vacmViewTreeFamilyType
MIN-ACCESS   read-only
DESCRIPTION  "Write access is not required."
```

```
OBJECT      vacmViewTreeFamilyStorageType
MIN-ACCESS   read-only
DESCRIPTION  "Write access is not required."
```

```
OBJECT      vacmViewTreeFamilyStatus
MIN-ACCESS   read-only
DESCRIPTION  "Create/delete/modify access to the
              vacmViewTreeFamilyTable is not required.
"
```

```
::= { vacmMIBCompliances 1 }
```

```
-- Units of conformance *****
```

```
vacmBasicGroup OBJECT-GROUP
```

```
  OBJECTS {
    vacmContextName,
    vacmGroupName,
    vacmSecurityToGroupStorageType,
    vacmSecurityToGroupStatus,
    vacmAccessContextMatch,
    vacmAccessReadViewName,
    vacmAccessWriteViewName,
    vacmAccessNotifyViewName,
    vacmAccessStorageType,
    vacmAccessStatus,
    vacmViewSpinLock,
    vacmViewTreeFamilyMask,
    vacmViewTreeFamilyType,
    vacmViewTreeFamilyStorageType,
    vacmViewTreeFamilyStatus
  }
```

```
  STATUS      current
```

```
  DESCRIPTION "A collection of objects providing for remote
              configuration of an SNMP engine which implements
```

```
        the SNMP View-based Access Control Model.  
    ::= { vacmMIBGroups 1 }
```

END

5. Intellectual Property

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

6. Acknowledgements

This document is the result of the efforts of the SNMPv3 Working Group. Some special thanks are in order to the following SNMPv3 WG members:

Harald Tveit Alvestrand (Maxware)
Dave Battle (SNMP Research, Inc.)
Alan Beard (Disney Worldwide Services)
Paul Berrevoets (SWI Systemware/Halcyon Inc.)
Martin Bjorklund (Ericsson)
Uri Blumenthal (IBM T.J. Watson Research Center)
Jeff Case (SNMP Research, Inc.)
John Curran (BBN)
Mike Daniele (Compaq Computer Corporation)
T. Max Devlin (Eltrax Systems)
John Flick (Hewlett Packard)
Rob Frye (MCI)
Wes Hardaker (U.C.Davis, Information Technology - D.C.A.S.)
David Harrington (Cabletron Systems Inc.)

Lauren Heintz (BMC Software, Inc.)
N.C. Hien (IBM T.J. Watson Research Center)
Michael Kirkham (InterWorking Labs, Inc.)
Dave Levi (SNMP Research, Inc.)
Louis A Mamakos (UUNET Technologies Inc.)
Joe Marzot (Nortel Networks)
Paul Meyer (Secure Computing Corporation)
Keith McCloghrie (Cisco Systems)
Bob Moore (IBM)
Russ Mundy (TIS Labs at Network Associates)
Bob Natale (ACE*COMM Corporation)
Mike O'Dell (UUNET Technologies Inc.)
Dave Perkins (DeskTalk)
Peter Polkinghorne (Brunel University)
Randy Presuhn (BMC Software, Inc.)
David Reeder (TIS Labs at Network Associates)
David Reid (SNMP Research, Inc.)
Aleksey Romanov (Quality Quorum)
Shawn Routhier (Epilogue)
Juergen Schoenwaelder (TU Braunschweig)
Bob Stewart (Cisco Systems)
Mike Thatcher (Independent Consultant)
Bert Wijnen (IBM T.J. Watson Research Center)

The document is based on recommendations of the IETF Security and Administrative Framework Evolution for SNMP Advisory Team. Members of that Advisory Team were:

David Harrington (Cabletron Systems Inc.)
Jeff Johnson (Cisco Systems)
David Levi (SNMP Research Inc.)
John Linn (Openvision)
Russ Mundy (Trusted Information Systems) chair
Shawn Routhier (Epilogue)
Glenn Waters (Nortel)
Bert Wijnen (IBM T. J. Watson Research Center)

As recommended by the Advisory Team and the SNMPv3 Working Group Charter, the design incorporates as much as practical from previous RFCs and drafts. As a result, special thanks are due to the authors of previous designs known as SNMPv2u and SNMPv2*:

Jeff Case (SNMP Research, Inc.)
David Harrington (Cabletron Systems Inc.)
David Levi (SNMP Research, Inc.)
Keith McCloghrie (Cisco Systems)
Brian O'Keefe (Hewlett Packard)
Marshall T. Rose (Dover Beach Consulting)

Jon Saperia (BGS Systems Inc.)
Steve Waldbusser (International Network Services)
Glenn W. Waters (Bell-Northern Research Ltd.)

7. Security Considerations

7.1. Recommended Practices

This document is meant for use in the SNMP architecture. The View-based Access Control Model described in this document checks access rights to management information based on:

- contextName, representing a set of management information at the managed system where the Access Control module is running.
- groupName, representing a set of zero or more securityNames. The combination of a securityModel and a securityName is mapped into a group in the View-based Access Control Model.
- securityModel under which access is requested.
- securityLevel under which access is requested.
- operation performed on the management information.
- MIB views for read, write or notify access.

When the User-based Access Control module is called for checking access rights, it is assumed that the calling module has ensured the authentication and privacy aspects as specified by the securityLevel that is being passed.

When creating entries in or deleting entries from the vacmViewTreeFamilyTable it is important to do such in the sequence as recommended in the DESCRIPTION clause of the vacmViewTreeFamilyTable definition. Otherwise unwanted access may be granted while changing the entries in the table.

7.2. Defining Groups

The groupNames are used to give access to a group of zero or more securityNames. Within the View-Based Access Control Model, a groupName is considered to exist if that groupName is listed in the vacmSecurityToGroupTable.

By mapping the combination of a securityModel and securityName into a groupName, an SNMP Command Generator application can add/delete securityNames to/from a group, if proper access is allowed.

Further it is important to realize that the grouping of <securityModel, securityName> tuples in the vacmSecurityToGroupTable does not take securityLevel into account. It is therefore important that the security administrator uses the securityLevel index in the vacmAccessTable to separate noAuthNoPriv from authPriv and/or authNoPriv access.

7.3. Conformance

For an implementation of the View-based Access Control Model to be conformant, it MUST implement the SNMP-VIEW-BASED-ACM-MIB according to the vacmMIBCompliance. It also SHOULD implement the initial configuration, described in appendix A.

7.4. Access to the SNMP-VIEW-BASED-ACM-MIB

The objects in this MIB control the access to all MIB data that is accessible via the SNMP engine and they may be considered sensitive in many environments. It is important to closely control (both read and write) access to these to these MIB objects by using appropriately configured Access Control models (for example the View-based Access Control Model as specified in this document).

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2578] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.
- [RFC2579] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.
- [RFC2580] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.
- [RFC3411] Harrington, D., Presuhn, R. and B. Wijnen, "An Architecture for describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.

- [SNMP3412] Case, J., Harrington, D., Presuhn, R. and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3412, December 2002.
- [RFC3414] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", STD 62, RFC 3414, December 2002.

8.2. Informative References

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

Appendix A - Installation

A.1. Installation Parameters

During installation, an authoritative SNMP engine which supports this View-based Access Control Model SHOULD be configured with several initial parameters. These include for the View-based Access Control Model:

1) A security configuration

The choice of security configuration determines if initial configuration is implemented and if so how. One of three possible choices is selected:

- initial-minimum-security-configuration
- initial-semi-security-configuration
- initial-no-access-configuration

In the case of a initial-no-access-configuration, there is no initial configuration, and so the following steps are irrelevant.

2) A default context

One entry in the vacmContextTable with a contextName of "" (the empty string), representing the default context. Note that this table gets created automatically if a default context exists.

vacmContextName	""
-----------------	----

3) An initial group

One entry in the vacmSecurityToGroupTable to allow access to group "initial".

vacmSecurityModel	3 (USM)
vacmSecurityName	"initial"
vacmGroupName	"initial"
vacmSecurityToGroupStorageType	anyValidStorageType
vacmSecurityToGroupStatus	active

4) Initial access rights

Three entries in the vacmAccessTable as follows:

- read-notify access for securityModel USM, securityLevel "noAuthNoPriv" on behalf of securityNames that belong to the group "initial" to the <restricted> MIB view in the default context with contextName "".
- read-write-notify access for securityModel USM, securityLevel "authNoPriv" on behalf of securityNames that belong to the group "initial" to the <internet> MIB view in the default context with contextName "".
- if privacy is supported, read-write-notify access for securityModel USM, securityLevel "authPriv" on behalf of securityNames that belong to the group "initial" to the <internet> MIB view in the default context with contextName "".

That translates into the following entries in the vacmAccessTable.

- One entry to be used for unauthenticated access (noAuthNoPriv):

vacmGroupName	"initial"
vacmAccessContextPrefix	""
vacmAccessSecurityModel	3 (USM)
vacmAccessSecurityLevel	noAuthNoPriv
vacmAccessContextMatch	exact
vacmAccessReadViewName	"restricted"
vacmAccessWriteViewName	""
vacmAccessNotifyViewName	"restricted"
vacmAccessStorageType	anyValidStorageType
vacmAccessStatus	active

- One entry to be used for authenticated access (authNoPriv) with optional privacy (authPriv):

vacmGroupName	"initial"
vacmAccessContextPrefix	""
vacmAccessSecurityModel	3 (USM)
vacmAccessSecurityLevel	authNoPriv
vacmAccessContextMatch	exact
vacmAccessReadViewName	"internet"
vacmAccessWriteViewName	"internet"
vacmAccessNotifyViewName	"internet"
vacmAccessStorageType	anyValidStorageType
vacmAccessStatus	active

5) Two MIB views, of which the second one depends on the security configuration.

- One view, the <internet> view, for authenticated access:
 - the <internet> MIB view is the following subtree:
"internet" (subtree 1.3.6.1)
- A second view, the <restricted> view, for unauthenticated access. This view is configured according to the selected security configuration:
- For the initial-no-access-configuration there is no default initial configuration, so no MIB views are pre-scribed.
- For the initial-semi-secure-configuration:

the <restricted> MIB view is the union of these subtrees:

- (a) "system" (subtree 1.3.6.1.2.1.1) [RFC3918]
- (b) "snmp" (subtree 1.3.6.1.2.1.11) [RFC3918]
- (c) "snmpEngine" (subtree 1.3.6.1.6.3.10.2.1) [RFC3411]
- (d) "snmpMPDStats" (subtree 1.3.6.1.6.3.11.2.1) [RFC3412]
- (e) "usmStats" (subtree 1.3.6.1.6.3.15.1.1) [RFC3414]

- For the initial-minimum-secure-configuration:

the <restricted> MIB view is the following subtree.
"internet" (subtree 1.3.6.1)

This translates into the following "internet" entry in the vacmViewTreeFamilyTable:

	minimum-secure	semi-secure
	-----	-----
vacmViewTreeFamilyViewName	"internet"	"internet"
vacmViewTreeFamilySubtree	1.3.6.1	1.3.6.1
vacmViewTreeFamilyMask	""	""
vacmViewTreeFamilyType	1 (included)	1 (included)
vacmViewTreeFamilyStorageType	anyValidStorageType	anyValidStorageType
vacmViewTreeFamilyStatus	active	active

In addition it translates into the following "restricted" entries in the vacmViewTreeFamilyTable:

	minimum-secure -----	semi-secure -----
vacmViewTreeFamilyViewName	"restricted"	"restricted"
vacmViewTreeFamilySubtree	1.3.6.1	1.3.6.1.2.1.1
vacmViewTreeFamilyMask	""	""
vacmViewTreeFamilyType	1 (included)	1 (included)
vacmViewTreeFamilyStorageType	anyValidStorageType	anyValidStorageType
vacmViewTreeFamilyStatus	active	active
vacmViewTreeFamilyViewName		"restricted"
vacmViewTreeFamilySubtree		1.3.6.1.2.1.11
vacmViewTreeFamilyMask		""
vacmViewTreeFamilyType		1 (included)
vacmViewTreeFamilyStorageType		anyValidStorageType
vacmViewTreeFamilyStatus		active
vacmViewTreeFamilyViewName		"restricted"
vacmViewTreeFamilySubtree		1.3.6.1.6.3.10.2.1
vacmViewTreeFamilyMask		""
vacmViewTreeFamilyType		1 (included)
vacmViewTreeFamilyStorageType		anyValidStorageType
vacmViewTreeFamilyStatus		active
vacmViewTreeFamilyViewName		"restricted"
vacmViewTreeFamilySubtree		1.3.6.1.6.3.11.2.1
vacmViewTreeFamilyMask		""
vacmViewTreeFamilyType		1 (included)
vacmViewTreeFamilyStorageType		anyValidStorageType
vacmViewTreeFamilyStatus		active
vacmViewTreeFamilyViewName		"restricted"
vacmViewTreeFamilySubtree		1.3.6.1.6.3.15.1.1
vacmViewTreeFamilyMask		""
vacmViewTreeFamilyType		1 (included)
vacmViewTreeFamilyStorageType		anyValidStorageType
vacmViewTreeFamilyStatus		active

B. Change Log

Changes made since RFC 2575:

- Removed reference from abstract as per RFC-Editor guidelines
- Updated references

Changes made since RFC 2275:

- Added text to vacmSecurityToGroupStatus DESCRIPTION clause to clarify under which conditions an entry in the vacmSecurityToGroupTable can be made active.
- Added REVISION clauses to MODULE-IDENTITY
- Clarified text in vacmAccessTable DESCRIPTION clause.
- Added a DEFVAL clause to vacmAccessContextMatch object.
- Added missing columns in Appendix A and re-arranged for clarity.
- Fixed oids in appendix A.
- Use the PDU Class terminology instead of RFC1905 PDU types.
- Added section 7.4 about access control to the MIB.
- Fixed references to new/revised documents
- Fix Editor contact information.
- fixed spelling errors
- removed one vacmAccessEntry from sample in appendix A.
- made some more clarifications.
- updated acknowledgement section.

Editors' Addresses

Bert Wijnen
Lucent Technologies
Schagen 33
3461 GL Linschoten
Netherlands

Phone: +31-348-480-685
EMail: bwijnen@lucent.com

Randy Presuhn
BMC Software, Inc.
2141 North First Street
San Jose, CA 95131
USA

Phone: +1 408-546-1006
EMail: randy_presuhn@bmc.com

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

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

Full Copyright Statement

Copyright (C) The Internet Society (2002). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

