SNMPv3

Overview – SNMPv3

- SNMPv3
  - Architecture
  - Message Format
  - Security
  - Applications
  - Textual Conventions
  - MIBs

SNMPv3 - RFCs

- RFC 3411, “An Architecture for Describing SNMP Management Frameworks”.
- RFC 3412, “Message Processing and Dispatching for the SNMP”.
- RFC 3413, “SNMP Applications”.

Section 1
SNMPv3 Architecture

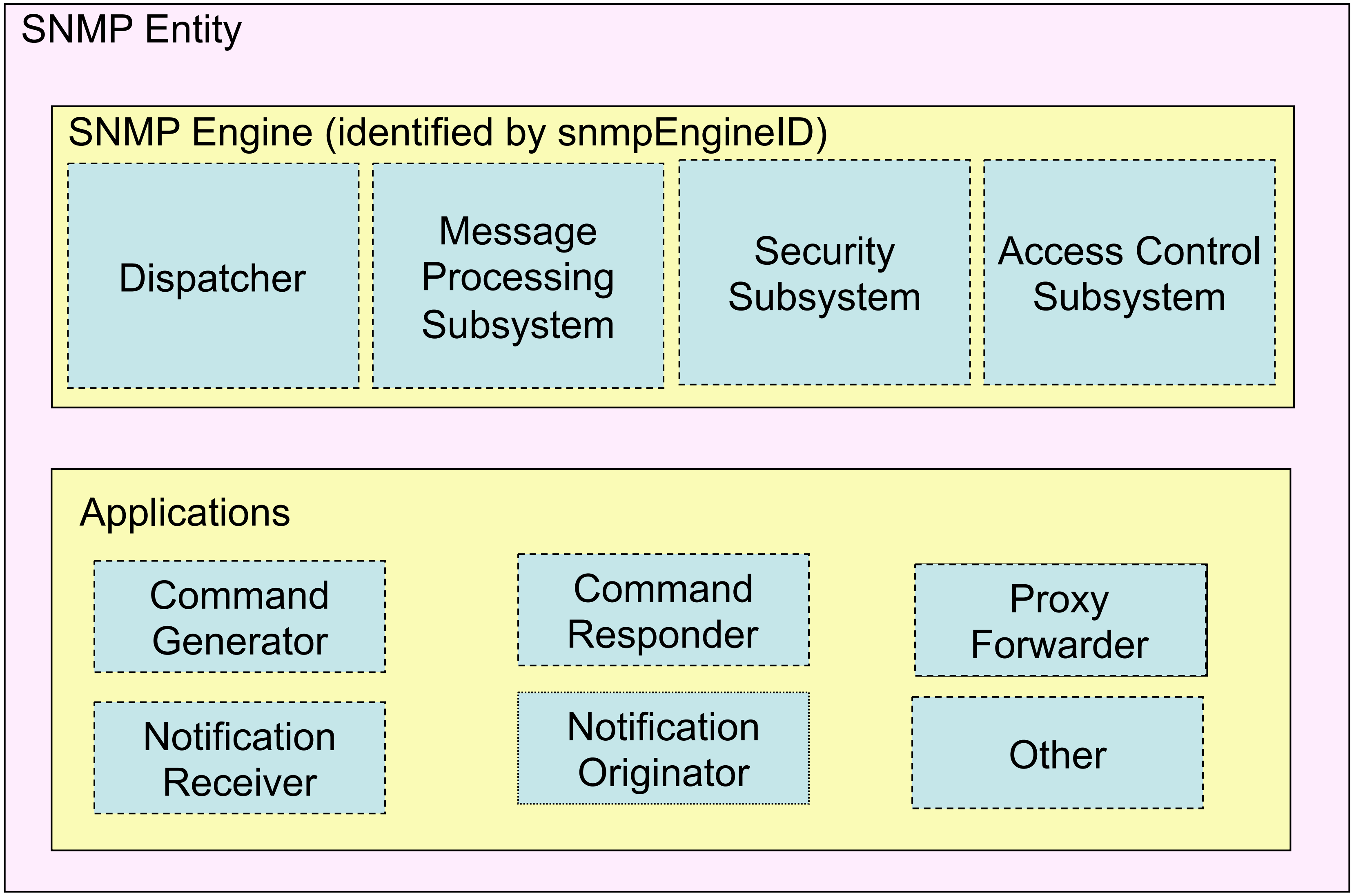
SNMPv3 Architecture

- SNMPv3 is SNMPv2 plus Administration and Security.
  - Proposes a modular architecture that can be easily extended.
  - Consists of :
    - SNMP entity
      - a SNMP engine
      - a set of applications
    - a new message format
    - a security model, and
    - an access control model.

SNMP Entity

- An SNMP entity contains an SNMP engine and one or more associated applications:
  - *command responder* and *notification originator* application
    - Access to management information
    - Agent function
  - *command generator* and/or *notification receiver* application
    - Manager function

SNMP Entity


SNMP Entity
SNMP Engine (identified by snmpEngineID)
Dispatcher
Message Processing Subsystem
Security Subsystem
Access Control Subsystem
Applications
Command Generator
Command Responder
Proxy Forwarder
Notification Receiver
Notification Originator
Other

SNMP Engine

- An SNMP Engine provides services for:
  - sending and receiving messages
  - authenticating and encrypting/decrypting messages, and
  - controlling access to managed objects.
- SNMP Engine consists of the following subsystems:
  - Dispatcher Subsystem
  - Message Processing Subsystem
  - Security Subsystem
  - Access Control Subsystem
- ‘snmpEngineID’ is the unique and unambiguous identifier of both an SNMP Entity and the SNMP Engine it contains.

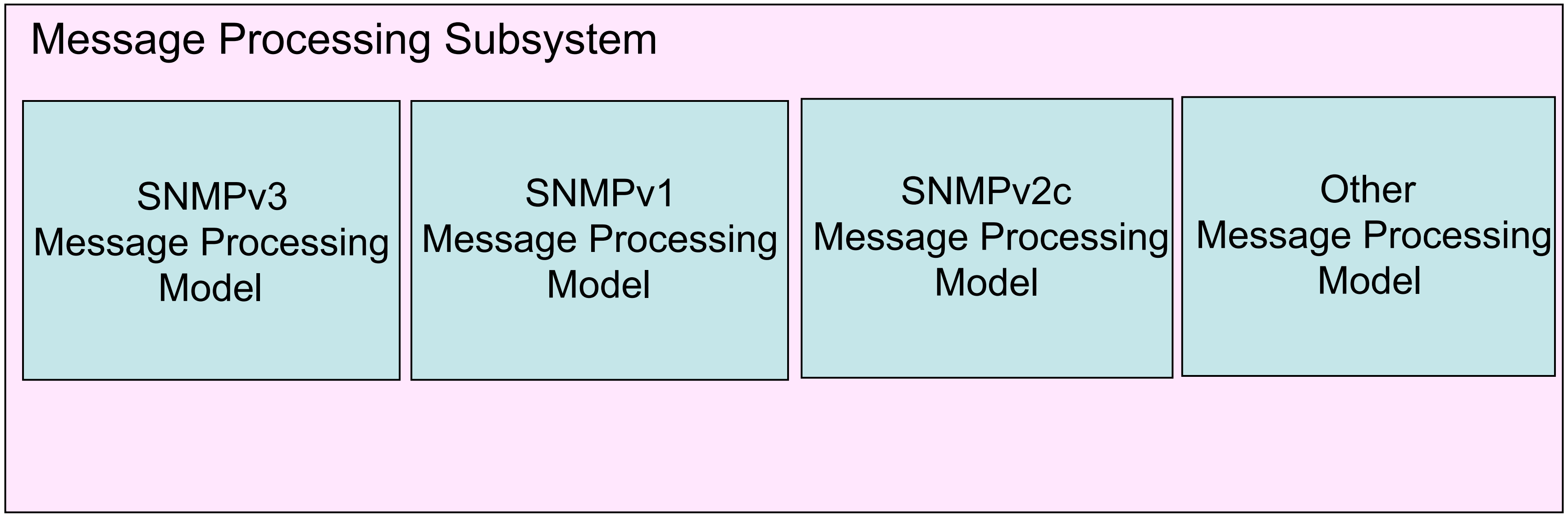
SNMP Engine - Dispatcher

- Allows for concurrent support of multiple versions of SNMP messages in the SNMP engine.
- It is responsible for
  - ✓ (1) accepting protocol data units (PDUs) from applications for transmission over the network and delivering incoming PDUs to applications;
  - ✓ (2) passing outgoing PDUs to the Message Processing Subsystem to prepare as messages, and passing incoming messages to the Message Processing Subsystem to extract the incoming PDUs;
  - ✓ (3) sending and receiving SNMP messages over the network.

SNMP Engine –
Message Processing Subsystem

- Responsible for:
  - preparing messages for sending, and
  - extracting data from received messages
- Contains multiple Message Processing Models
- Each Message Processing Model is specific to a particular version of an SNMP message and prepares or extracts version-specific message formats.

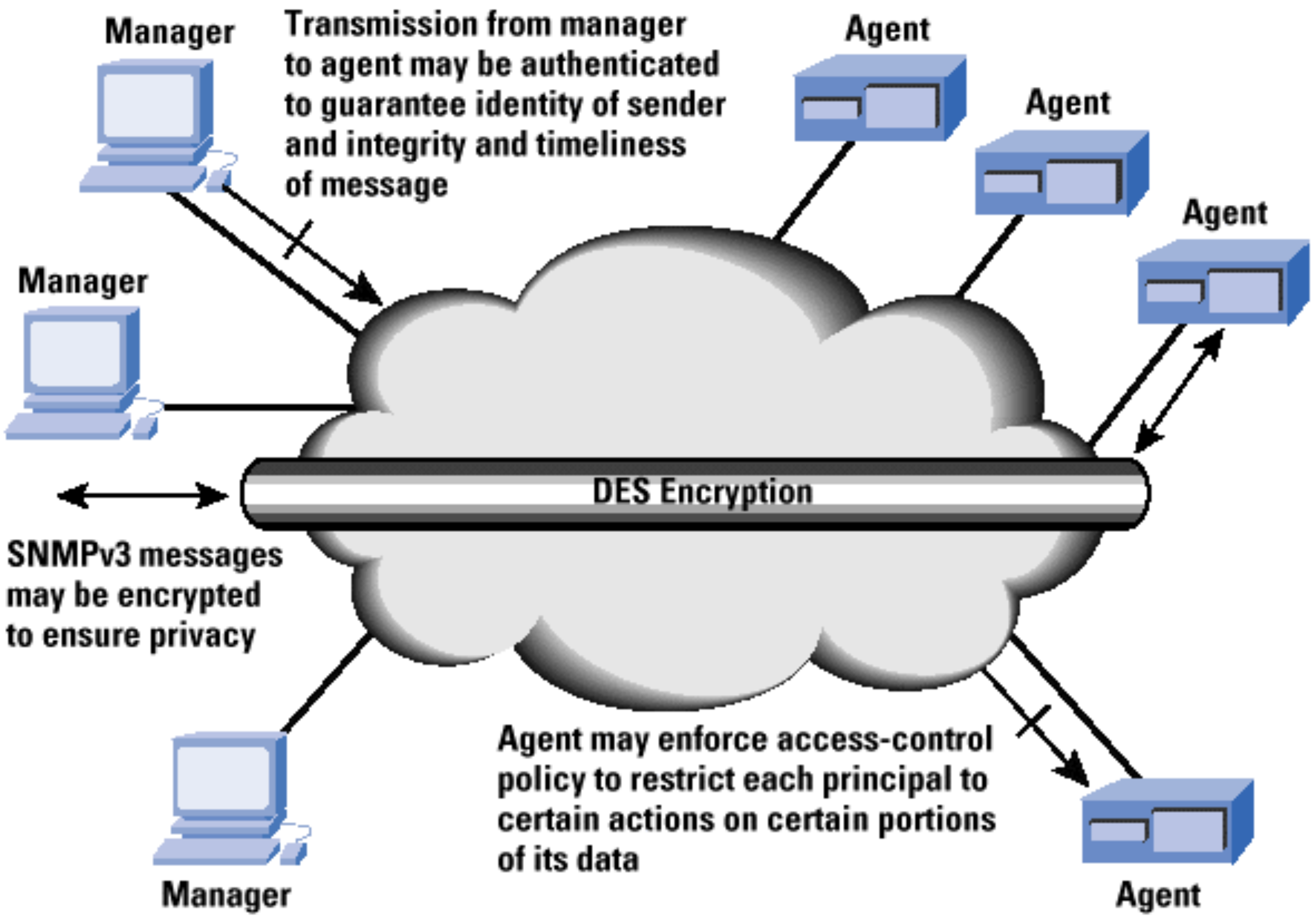
SNMP Engine –
Message Processing Subsystem


Message Processing Subsystem
SNMPv3 Message Processing Model
SNMPv1 Message Processing Model
SNMPv2c Message Processing Model
Other Message Processing Model

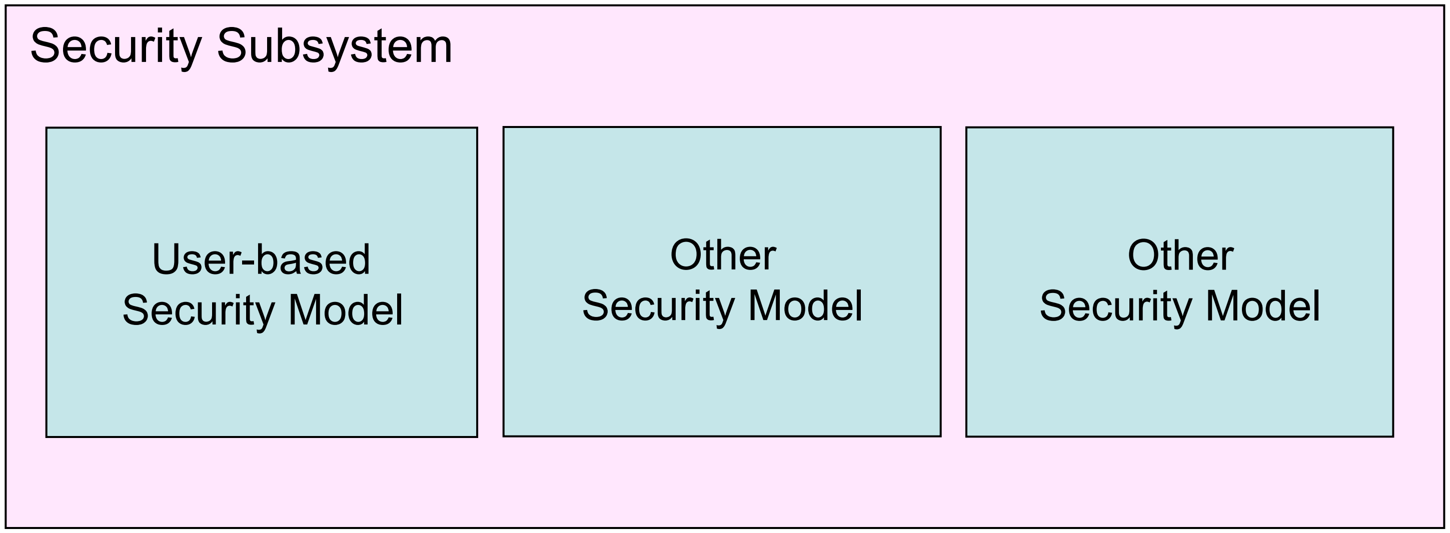
SNMP Engine – Security Subsystem

- The Security Subsystem provides security services such as:
  - Authenticating messages, and
  - Encrypting/decrypting messages for privacy
- Contains one or more Security Models
- A **Security Model** specifies
  - the threats against which it protects, and
  - the security protocols used to provide security services such as authentication and privacy.

SNMP Engine – Security Subsystem


Manager
Transmission from manager to agent may be authenticated to guarantee identity of sender and integrity and timeliness of message
Agent
Agent
Agent
Manager
DES Encryption
SNMPv3 messages may be encrypted to ensure privacy
Agent may enforce access-control policy to restrict each principal to certain actions on certain portions of its data
Manager
Agent

SNMP Engine – Security Subsystem



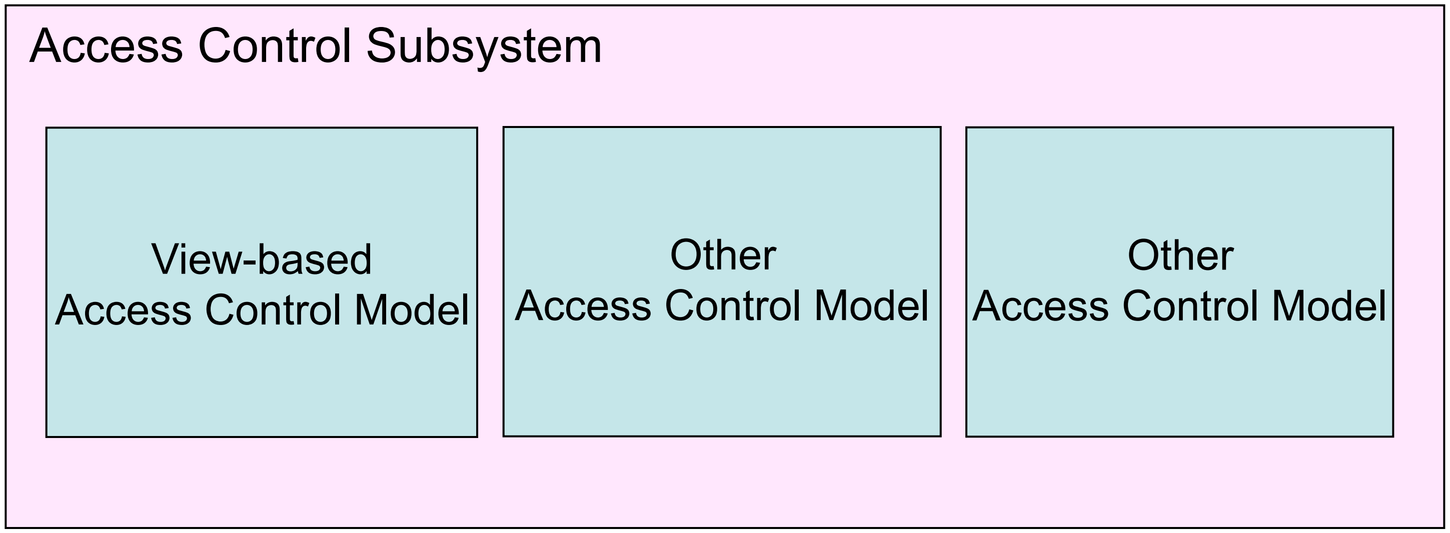
SNMPv3 Security Level

- Every message has an associated <u>security level</u>.
- SNMPv3 recognizes three levels of security:
  - without authentication and without privacy (noAuthNoPriv)
  - with authentication but without privacy (authNoPriv)
  - with authentication and with privacy (authPriv)

SNMP Engine – Access Control Subsystem

- The Access Control Subsystem provides authorization services by means of one or more Access Control Models.
- An **Access Control Model** defines a particular access decision function for computing access rights.

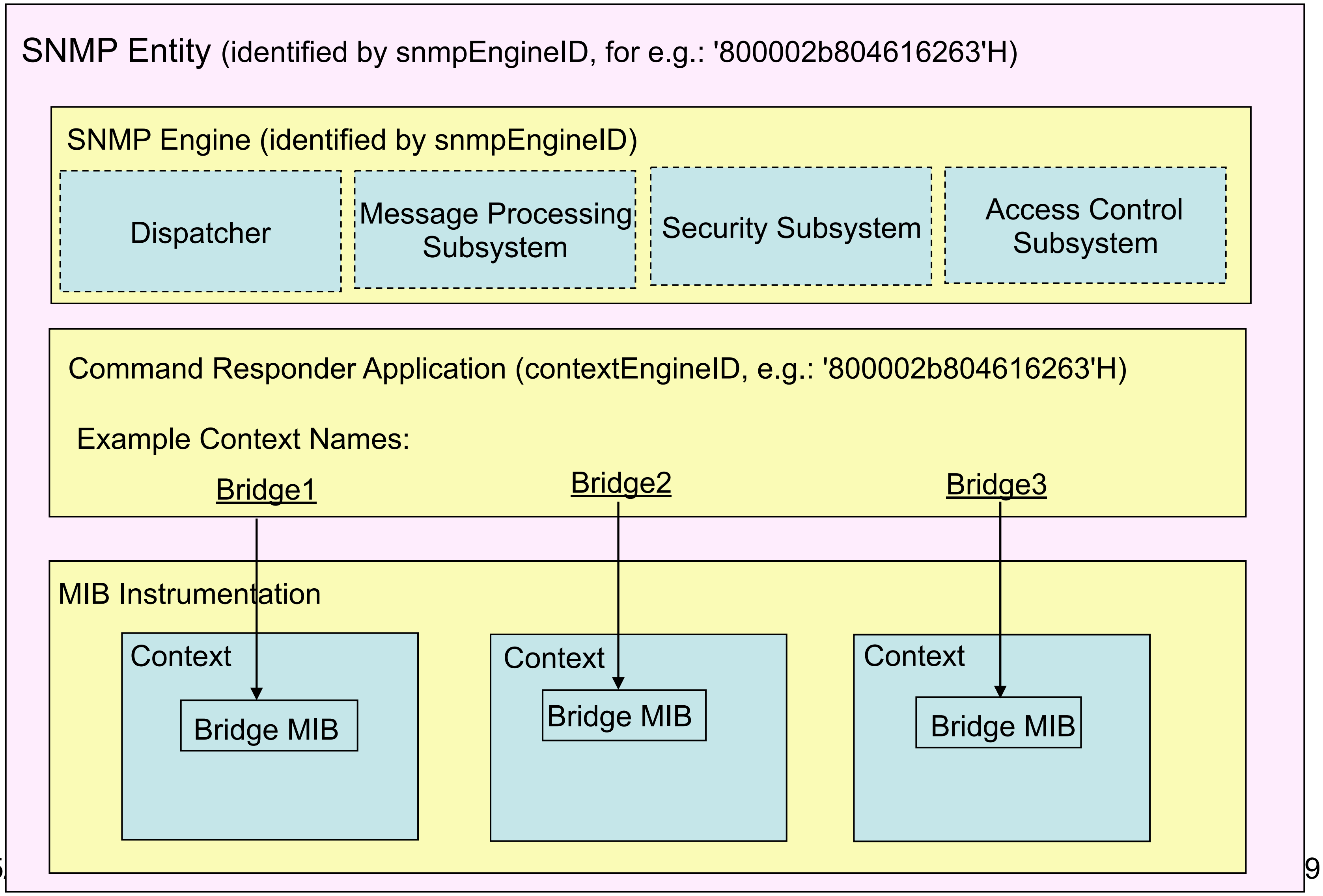
SNMP Engine – Access Control Subsystem


Access Control Subsystem
View-based Access Control Model
Other Access Control Model
Other Access Control Model

SNMP Context

- An SNMP context is a collection of management information
  - An item of management information may exist in more than one context.
  - An SNMP entity potentially has access to many contexts.
- Often a context is a physical or a logical device and is defined as a subset of a single SNMP entity.
- The combination of a **contextEngineID** and a **contextName** unambiguously identifies a context.
- To identify an individual item of management information, its **contextName** and **contextEngineID** must be identified in addition to its object type and its instance.

SNMP Context

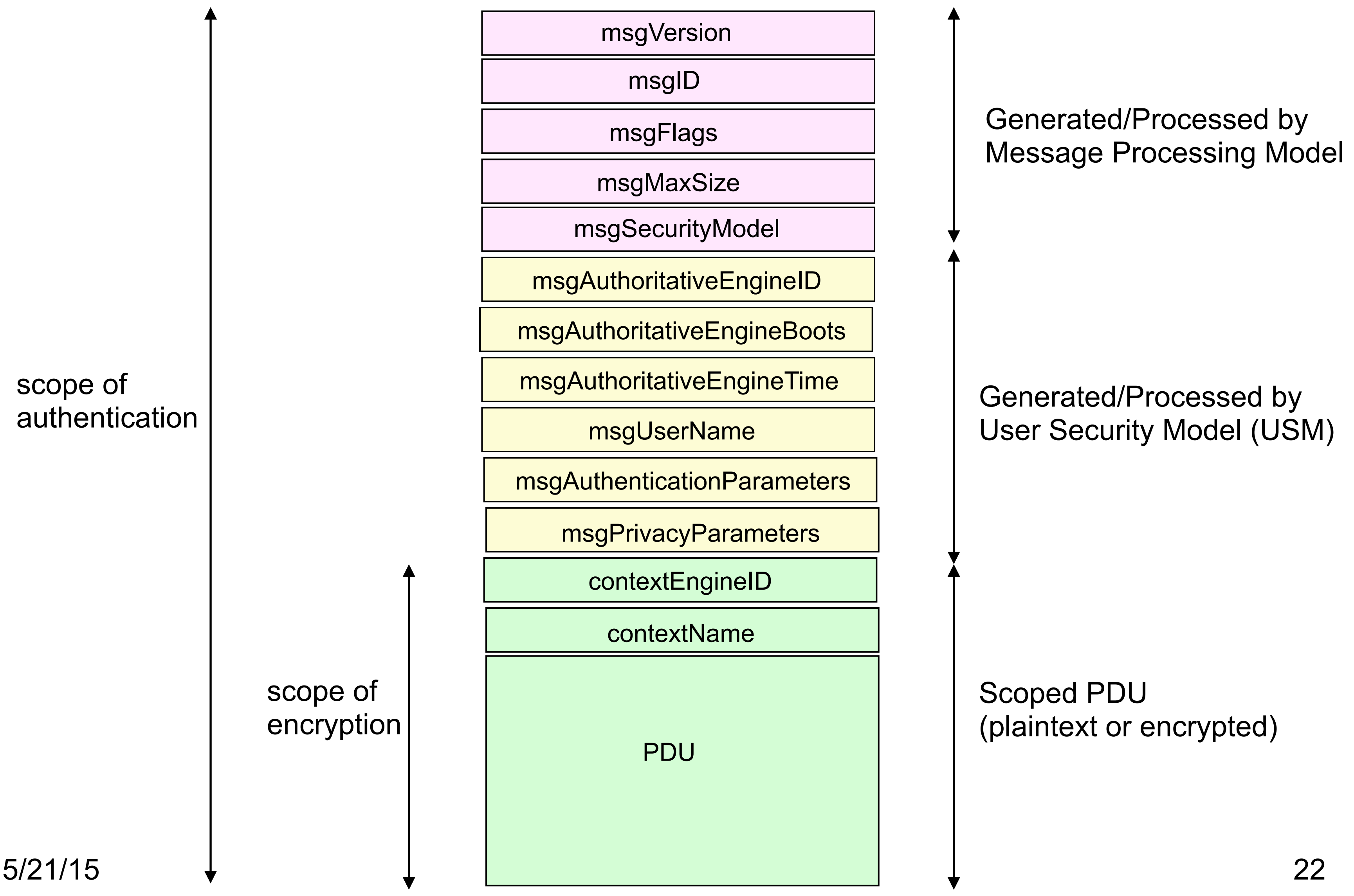

SNMP Entity (identified by snmpEngineID, for e.g.: '800002b804616263'H)
SNMP Engine (identified by snmpEngineID)
Dispatcher
Message Processing Subsystem
Security Subsystem
Access Control Subsystem
Command Responder Application (contextEngineID, e.g.: '800002b804616263'H)
Example Context Names:
Bridge1
Bridge2
Bridge3
MIB Instrumentation
Context
Bridge MIB
Context
Bridge MIB
Context
Bridge MIB

Section 2
SNMPv3 Message Format

SNMPv3 – Message Format

- Contains
  - Message Header
    - msgVersion
    - msgID
    - msgMaxSize
    - msgFlags
    - msgSecurityModel
  - Security Header
  - SNMPv2 PDU (plain text or encrypted)

SNMPv3 Message Format



msgVersion
msgID
msgFlags
msgMaxSize
msgSecurityModel
msgAuthoritativeEngineID
msgAuthoritativeEngineBoots
msgAuthoritativeEngineTime
msgUserName
msgAuthenticationParameters
msgPrivacyParameters
contextEngineID
contextName
PDU
scope of authentication
scope of encryption
Generated/Processed by Message Processing Model
Generated/Processed by User Security Model (USM)
Scoped PDU (plaintext or encrypted)

SNMPv3 – Message Format

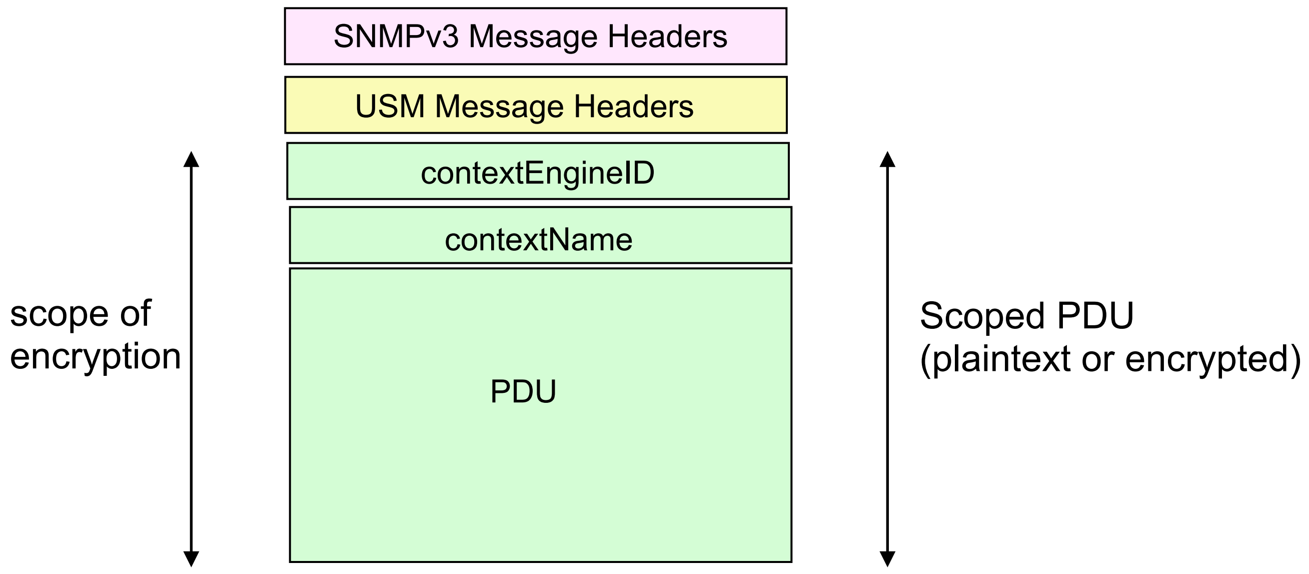
- msgVersion
  - A value of 3 identifies the version of the message as an SNMPv3 message
- msgID
  - Used to coordinate request and response messages between two SNMP entities
  - Similar to the reqID used within a PDU
- msgMaxSize
  - Indicates the maximum message size that the sender can support
  - Used to determine how big a response to a request message can be
  - Values range from 484 to 2^{31}-1

SNMPv3 – Message Format

- msgFlags
  - Indicates the security level that the sender has applied to the message.
  - The 3 bits defined are reportableFlag, authFlag, and the privFlag
- msgSecurityModel
  - An integer value which identifies the message security model that the sender used to generate the message.
  - The receiver must use the same security model to perform security processing for the message.

SNMP ScopedPDU

- A scopedPDU is a block of data containing a contextEngineID, a contextName, and a PDU.



SNMPv3 – Reports

- Used for engine-to-engine communication
- Processed by the SNMPv3 message processing model
- Allows to report the following type of errors (to the sender) when processing a SNMP message:
  - A response message could not be generated
  - A message was received with the authFlag cleared and the privFlag set
  - An error occurred while providing authentication or privacy services for an incoming message

SNMPv3 – Reports

type	reqid	0	0	Variable bindings

- The PDU type 0xA8 indicates a Report PDU
- ‘reqid’ consists of request identifier of the message that triggered the Report
- Variable bindings will contain a single object identifier and its value. Used to determine the problem that the report is identifying.

Section 3
SNMPv3 Security

SNMPv3 Security Requirements

1. Has the message been altered?
2. Is the message coming from a valid user?
3. Has the message been delayed or replayed?
4. Can sensitive information be protected against eavesdroppers?
5. Is the user allowed to access MIB objects specified in the message?

SNMPv3 Security Requirements

The **principal threats** against which a Security Model should provide protection include:

- **Modification of Information**

  The modification threat of in-transit SNMP messages by an unauthorized entity to effect unauthorized management operations.

- **Masquerade**

  Unauthorized management operations may be attempted by assuming the identity of someone who has appropriate authorizations.

SNMPv3 Security Requirements

The **secondary threats** against which a Security Model should provide protection include:

- **Message Stream Modification**

  Messages may be re-ordered, delayed or replayed to an extent to effect unauthorized management operations.
- **Disclosure**

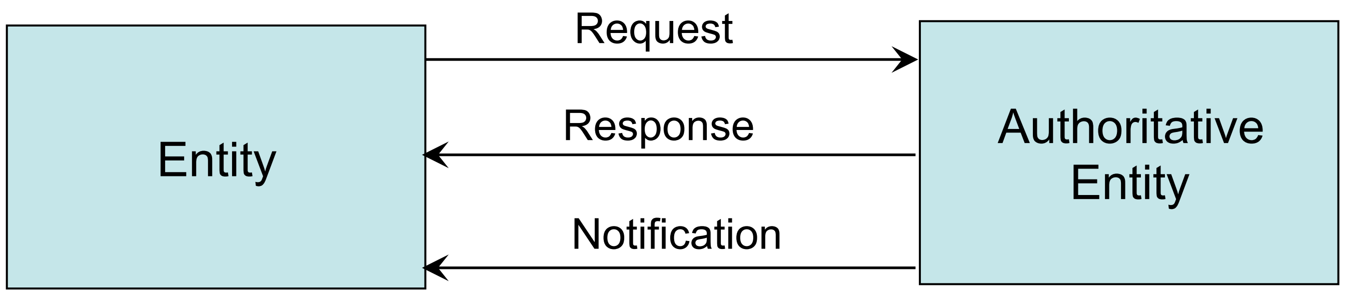
  Eavesdropping on the exchanges between SNMP engines.

SNMPv3 – Security

- Authoritative and Non-authoritative Entities
- Security Parameters
- Discovery of snmpEngineID
- Message Timeliness
- Key Management
- USM MIB

Authoritative & Non-Authoritative Entities

- Authoritative entity is the one which
  - receives and responds to requests
  - generates notifications



SNMPv3 – Security Parameters

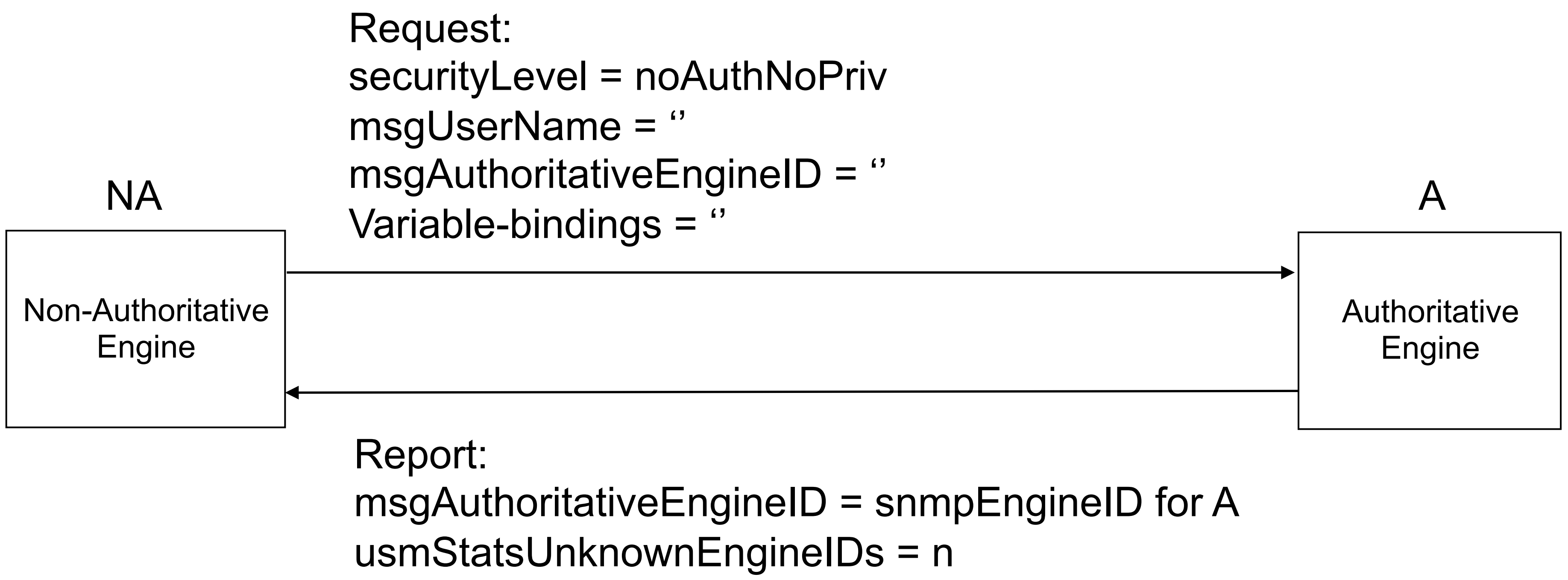
- Every SNMPv3 message contains security parameters.
- The following are USM security parameters:

```
UsmSecurityParameters ::=
    SEQUENCE {
        msgAuthoritativeEngineID          OCTET STRING
        msgAuthoritativeEngineBoots       INTEGER
        msgAuthoritativeEngineTime        INTEGER
        msgUserName                       OCTET STRING
        msgAuthenticationParameters       OCTET STRING
        msgPrivacyParameters              OCTET STRING
   }
```

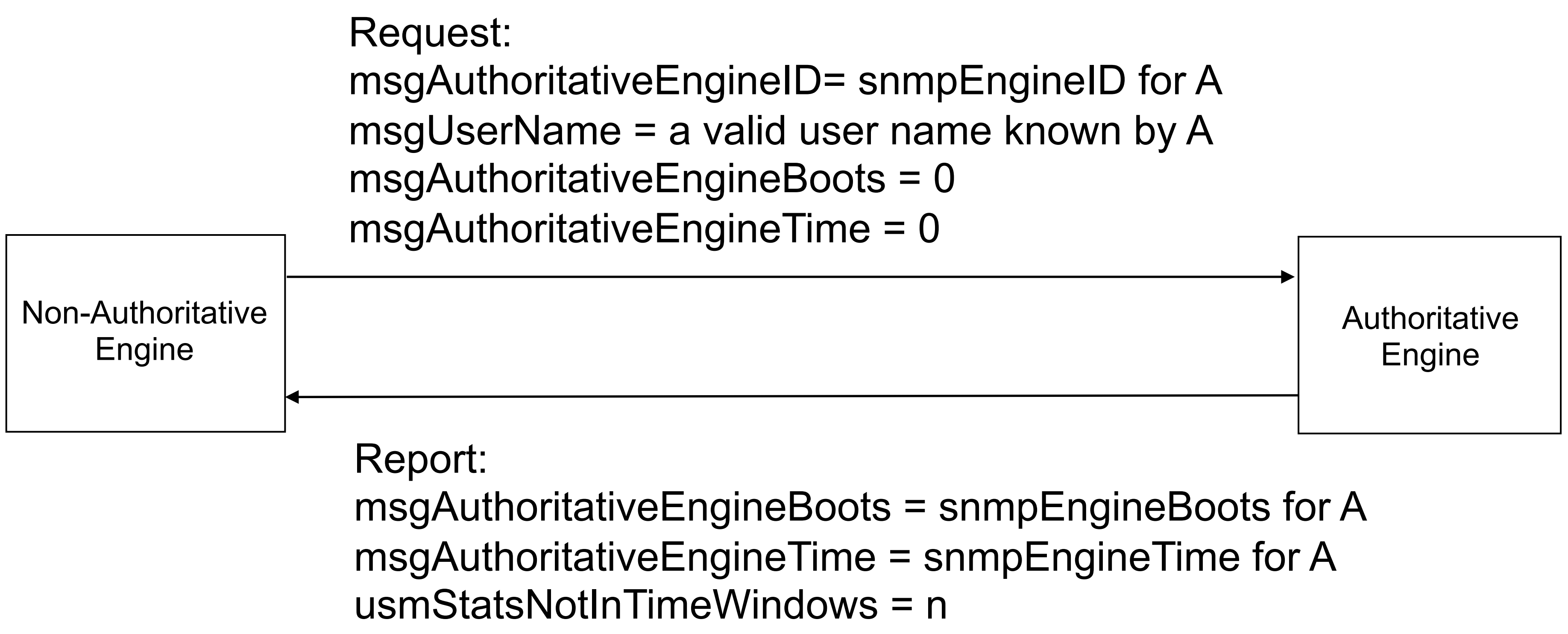
SNMPv3 – EngineID Discovery

- USM mechanism allows a non-authoritative SNMP entity to learn the snmpEngineID of the authoritative engine.
- The non-authoritative entity sends a Request message to the authoritative entity with
  - A *securityLevel* of “noAuthNoPriv”
  - A *msgUserName* of zero length
  - A *msgAuthoritativeEngineID* of zero length
  - A empty variable-bindings
- The authoritative entity responds by sending a Report message with its snmpEngineID value in the msgAuthoritativeEngineID field

SNMPv3 – EngineID Discovery


Request:
securityLevel = noAuthNoPriv
msgUserName = ‘’
msgAuthoritativeEngineID = ‘’
Variable-bindings = ‘’
NA
A
Non-Authoritative Engine
Authoritative Engine
Report:
msgAuthoritativeEngineID = snmpEngineID for A
usmStatsUnknownEngineIDs = n

SNMPv3 - EngineBoots & EngineTime Discovery


Request:
msgAuthoritativeEngineID= snmpEngineID for A
msgUserName = a valid user name known by A
msgAuthoritativeEngineBoots = 0
msgAuthoritativeEngineTime = 0
Non-Authoritative Engine
Authoritative Engine
Report:
msgAuthoritativeEngineBoots = snmpEngineBoots for A
msgAuthoritativeEngineTime = snmpEngineTime for A
usmStatsNotInTimeWindows = n

SNMPv3 – Timeliness

- An authoritative engine will discard a message if:
  - $boots_A = 2147483647$, or
  - $boots_{NA}$ and $boots_A$ are not equal, or
  - $(boots_{NA} = boots_A)$ & $|time_A – time_{NA}| > 150s$)

SNMPv3 – Timeliness

- When a non-authoritative engine receives a message
  - $boots_A$ and $time_A$ are determined from the msgAuthoritativeEngineBoots and msgAuthoritativeEngineTime fields
  - If ($boots_A > boots_{NA}$) or ($boots_A = boots_{NA}$ & $time_A > time_{NA}$) is true, then the non-authoritative engine will update its $boots_{NA}$ and $time_{NA}$ values.
- A non-authoritative engine will discard a message if:
  - $boots_{NA} = 2147483647$
  - $boots_{NA} > boots_A$
  - ($boots_{NA} = boots_A$) & (($time_{NA} – time_A$) > 150s)

SNMPv3 – Key Management

- User name is mapped to a textual password:
  - USM has proposed procedures to convert a password into
    1. a 16-byte OCTET STRING that can be used by the HMAC-MD5-96 authentication protocol, and
    2. a 20-byte OCTET STRING that can be used by the HMAC-SHA-96 authentication protocol.
- A single password for each user irrespective of the number of SNMP entities in a network
- Localized secret keys will be derived for a user for each authoritative entity.

SNMPv3 – Password to Key Generation

1) Take the user's password as input and produce a string of length 220 octets to form digest0.
2) If a 16-octet key is desired, take the MD5 hash of digest0 to form digest1.
3) If a 20-octet key is desired, take the SHA-1 hash of digest0 to form digest1.
4) The output is user's key.

SNMPv3 – Localized Key Generation

1) Form the string digest2 by concatenating digest1, the authoritative engine's **snmpEngineID** value, and digest1.
2) If a 16-octet key is desired, take the MD5 hash of digest2.
3) If a 20-octet key is desired, take the SHA-1 hash of digest2.
4) The output is user's localized key for an authoritative SNMP entity.

SNMPv3 – Key Management (contd.)

- Key values can be changed through SNMP
- Steps involved in a remote key change operation:
  - Generate a new password for a user
  - Get the new localized key value
  - Generate a 'random' 16-byte OCTET STRING
  - Calculate a value 'delta' using previous and new key values:

```
–          temp <- MD5 (oldkey + random)
–          for (i = 0; i < 16; i++) {
–                delta [i] = (temp [i])  XOR  (newkey [i])
–          }
```

  - Concatenate 'random' and 'delta' to form a 32-byte OCTET STRING, which will be used to remotely configure an authoritative entity's new secret key.

SNMPv3 – Key Management (contd.)

- A remote entity will calculate the new key value given ‘random’ and ‘delta’:

  Note: if (a XOR b = c) then (a XOR c = b))

```
temp <- MD5 (oldkey + random)
for (i=0; i < 16; i++)
{
    newkey [i] = temp [i]  XOR  delta[i]
}
```

SNMPv3 – Authentication MD5

- 16-byte secret authentication key
- Create ‘extendedKey’ which is the secret authentication key appended with 48 zero octets.
- The 12-octet MAC is computed as follows:

```
temp <- MD5 (K2 + MD5 (K1 + message)) // 16-byte octet string
MAC <- temp[0…11]
```

where K1 and K2 are computed as follows:

```
for (i = 0; i < 64; i++) {
        K1 [i] = extendedKey [i] XOR 0x36
}
for (i = 0; i < 64; i++) {
        K2 [i] = extendedkey [i] XOR 0x5C
}
```

SNMPv3 – Authentication SHA

- 20-byte secret authentication key
- Create ‘extendedKey’ which is the secret auth key appended with 44 zero octets.
- The 12-octet MAC is computed as follows:

```
temp <- SHA (K2 + SHA (K1 + message)) // 16-byte octet string
MAC <- temp[0…11]
where K1 and K2 are computed as follows:
for (i = 0; i < 64; i++) {
        K1 [i] = extendedKey [i] XOR 0x36
}
for (i = 0; i < 64; i++) {
        K2 [i] = extendedkey [i] XOR 0x5C
}
```

SNMPv3 – Privacy CBC-DES

- Symmetric secret key algorithm
- 16-byte secret key
- ‘msgPrivacyParameters’ field will contain an 8-byte octet string that represents a ‘salt’ value for the initialization vector used in DES encryption.

Section 4
SNMPv3 Applications

SNMP Applications

- SNMP applications make use of the services provided by the SNMP engine
- The different types of applications include:
  - **command generators** monitor and manipulate mgmt data
  - **command responders** provide access to management data,
  - **notification originators** initiate asynchronous messages,
  - **notification receivers** process asynchronous messages, and
  - **proxy forwarders** forward messages between entities.

Command Generator Applications

- Initiate SNMP Get, Get-Next, Get-Bulk, and Set requests and process responses to those requests
- The request identifier from the response PDU should match what was used in the original request.
- The response values of msgVersion, securityModel, contextEngineID, and contextName must match original request values.

Command Responder Applications

- Provide access to management data.
- Process a request when the contextEngineID specified matches the local SNMP engine.
- Perform appropriate operation and generate a response.
- Each object in the request’s variable binding is tested to see if it can be accessed based on the MIB view.

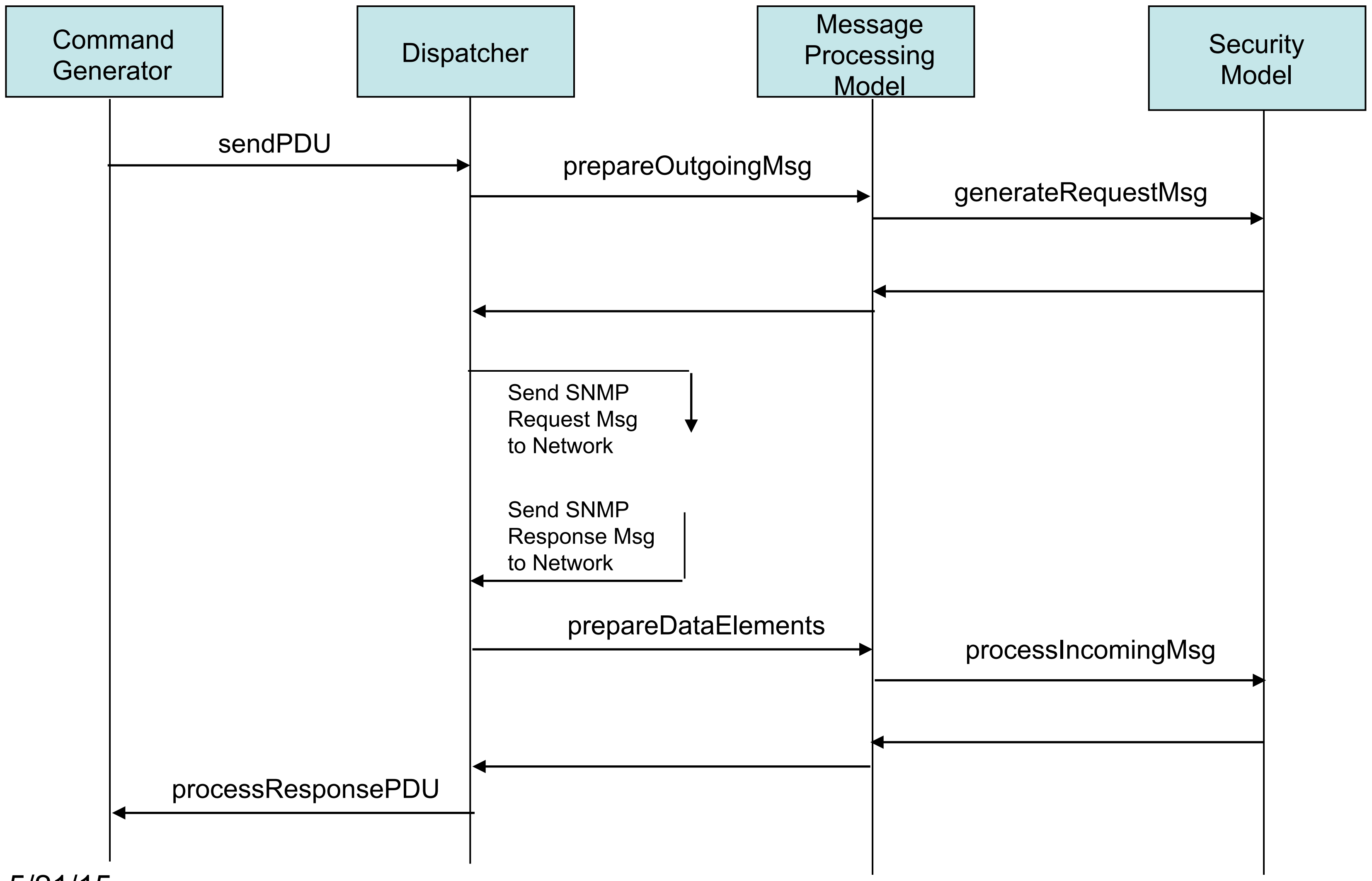
Notification Originator Application

- Generates SNMPv2-Trap or Inform notification.
- SNMPv3 defines MIB tables to configure the following:
  - target address(es) to send notifications to,
  - whether a SNMPv2-Trap or Inform PDU is sent, and
  - what timeout and retry counts to use for Inform PDUs.

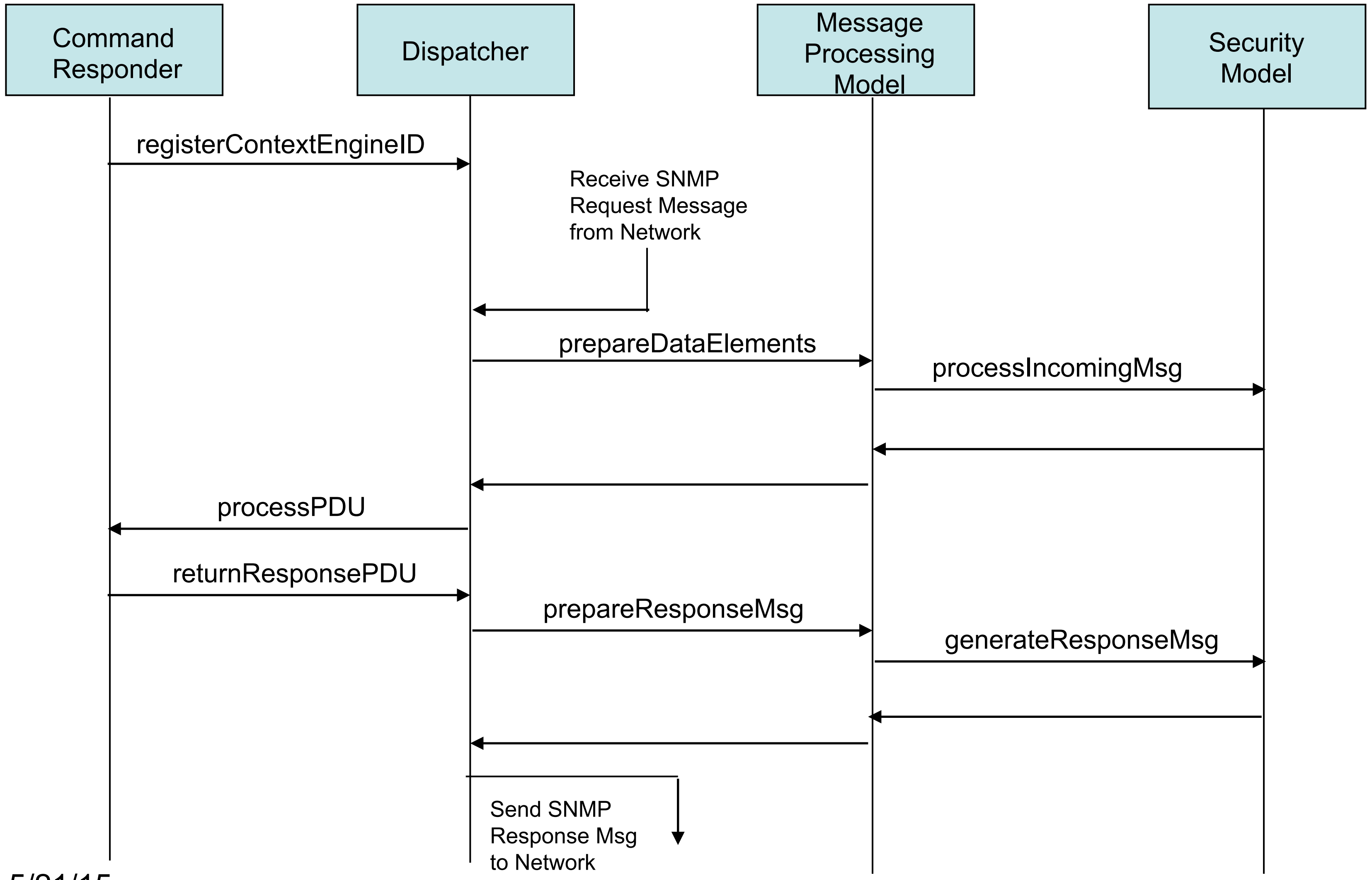
Notification Receiver Application

- Registers with the Dispatcher to receive SNMPv2-Trap and/or Inform PDUs.
- When Inform PDU is received, a response is generated using the original request identifier and variable-bindings, setting the error-status and error-index to zero.

SNMPv3 Command Generator Call Flow



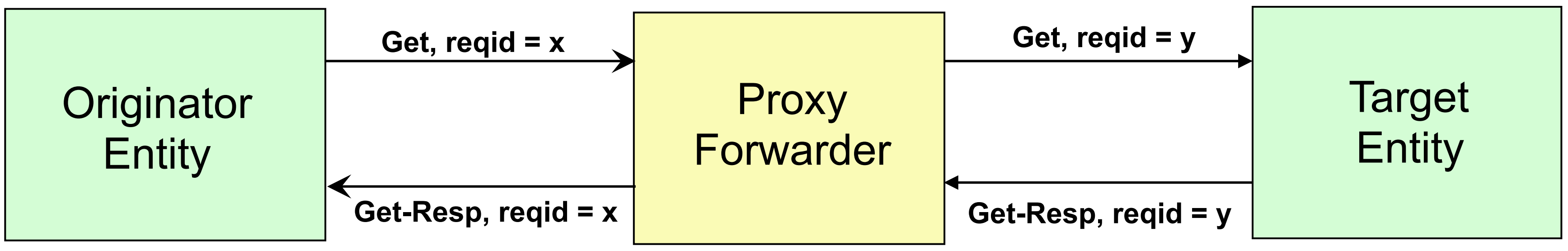
SNMPv3 Command Responder Call Flow

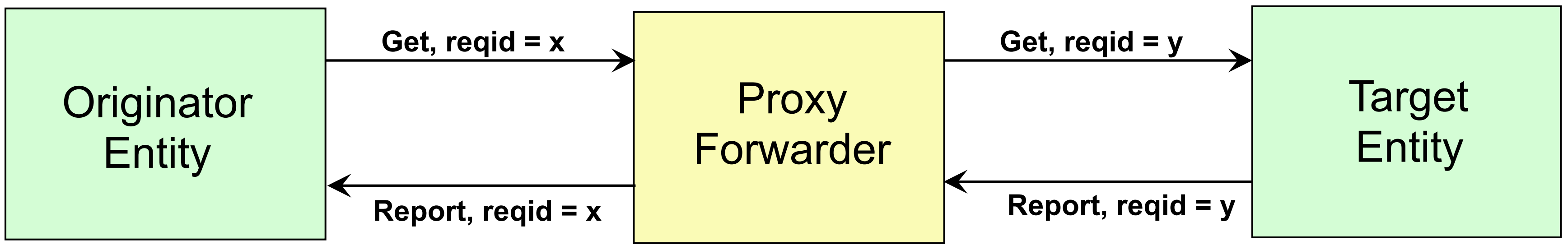


Proxy Forwarder Application

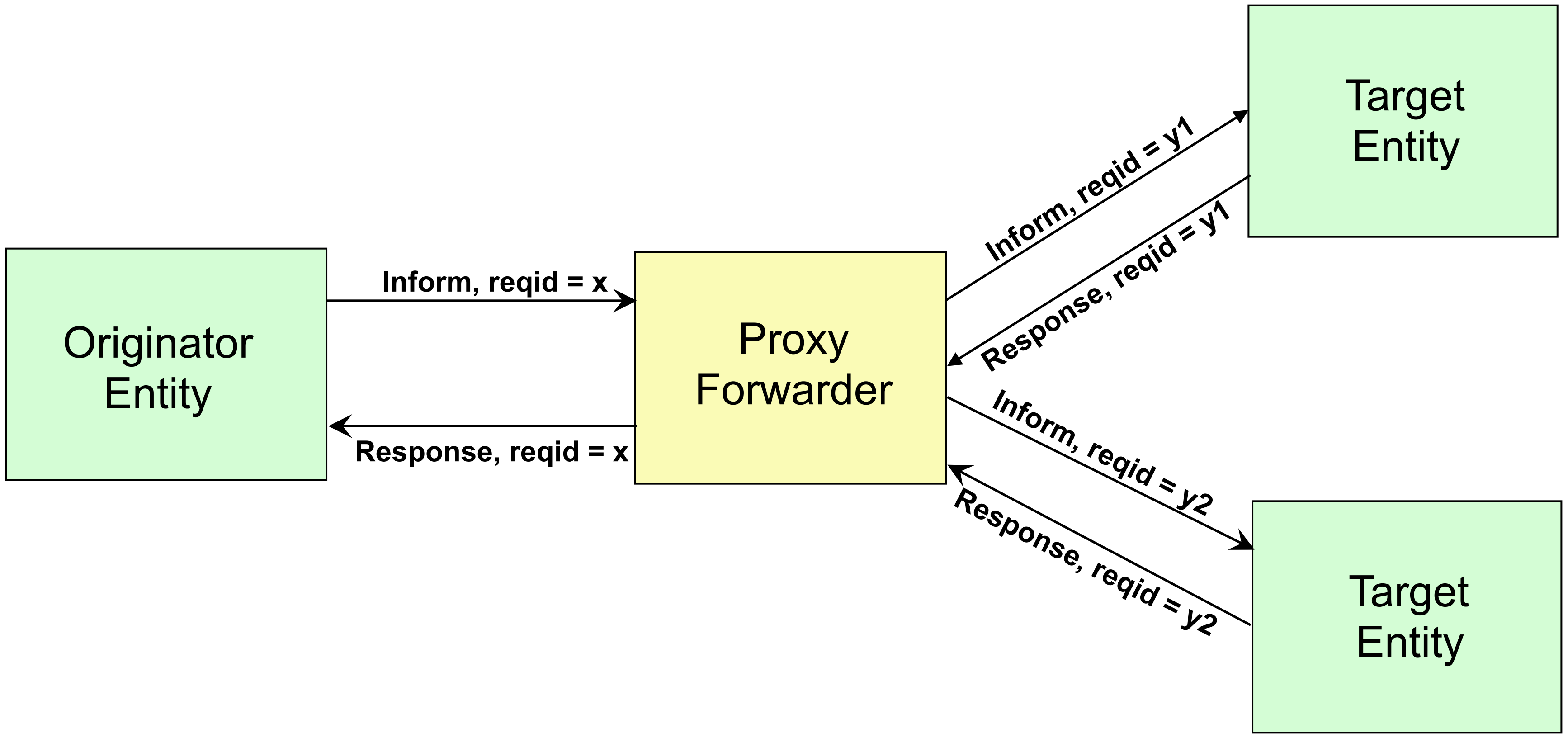
- Forwards requests and/or notifications to other SNMP entities.
- Forwards responses to the SNMP entity that sent the original message.
- SNMPv3 defines MIB tables on how to map incoming message to one or more target addresses.

Proxy Forwarder Application





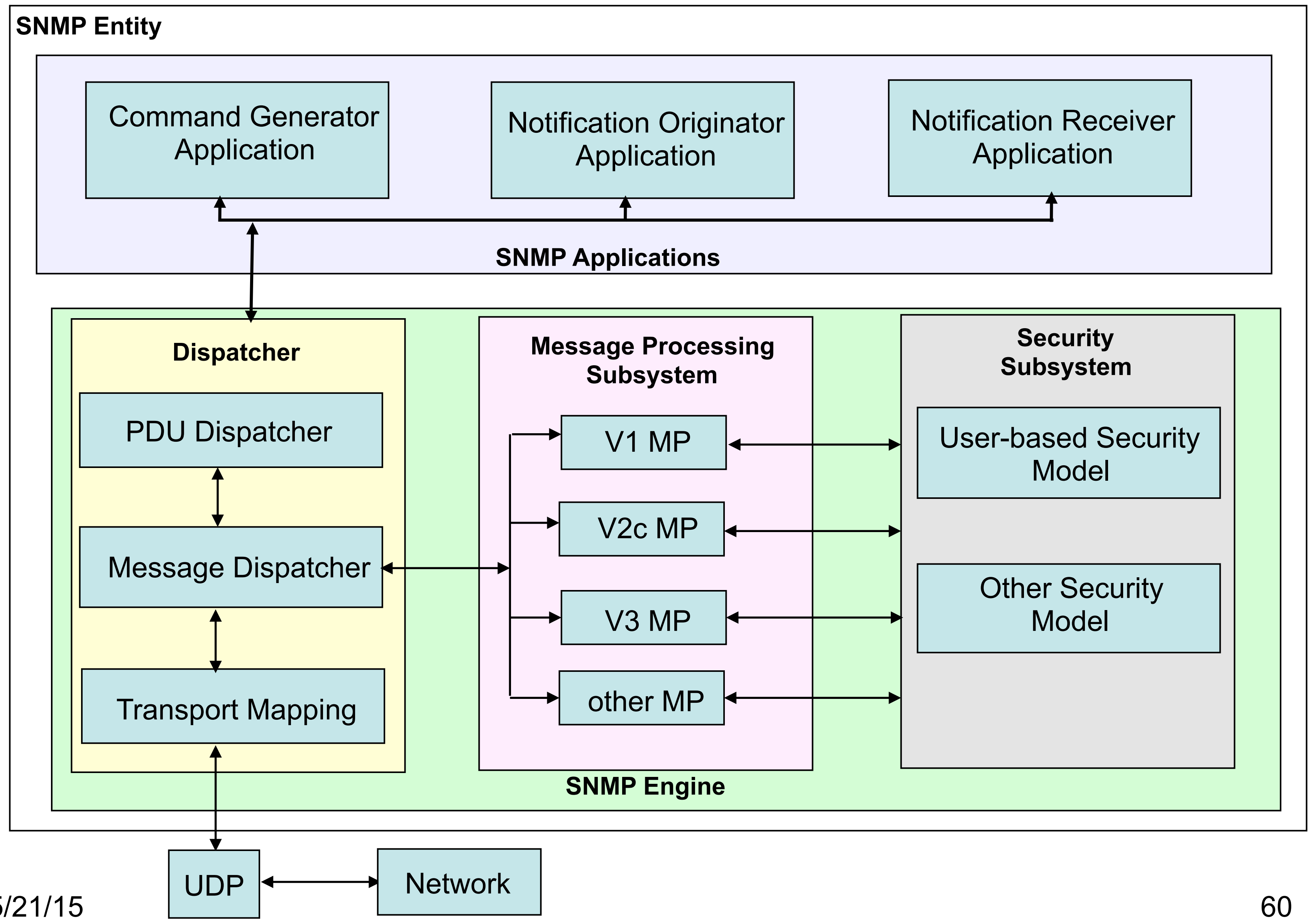
Proxy Forwarder Application



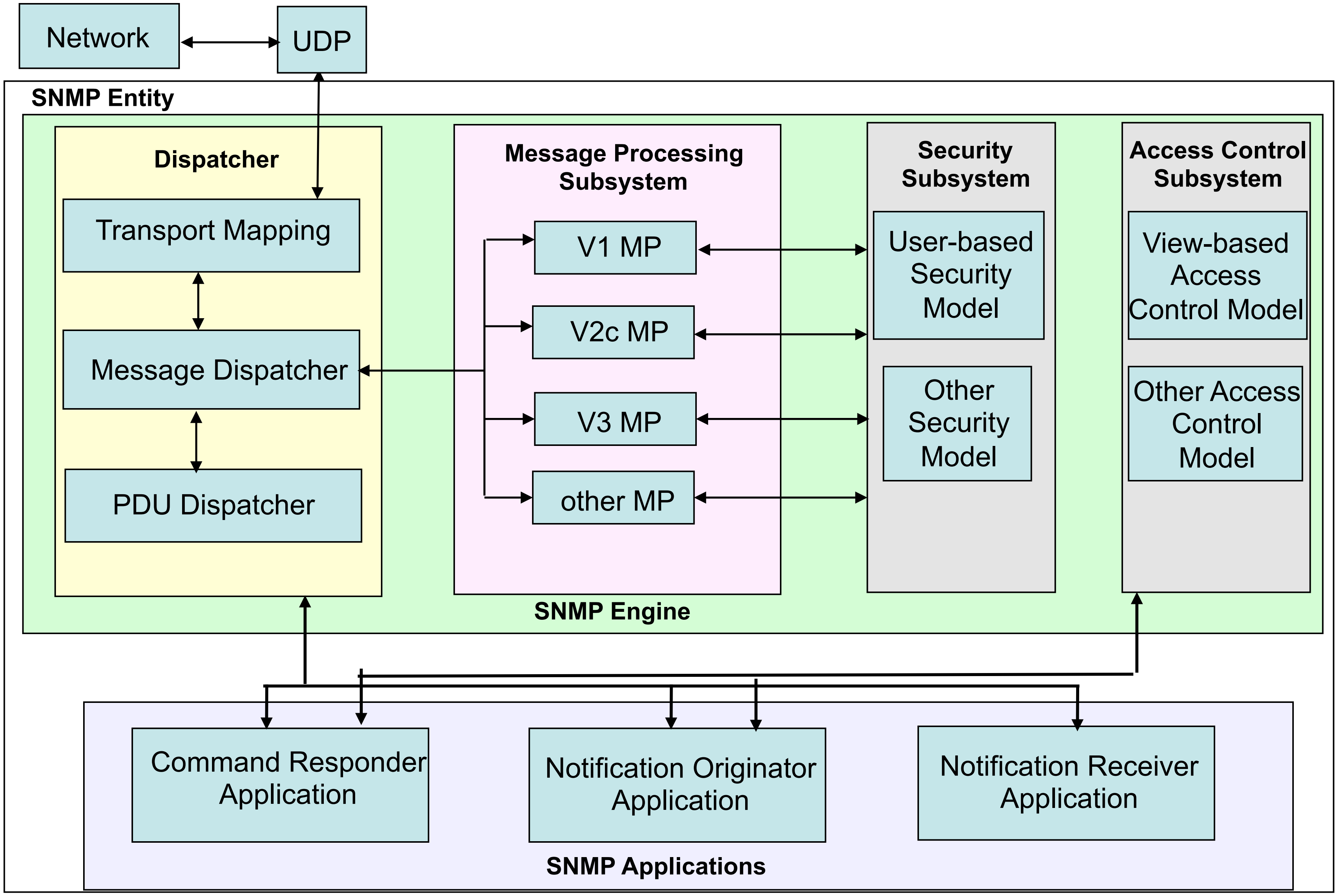
SNMPv3 Implementations

- Traditional SNMP agent: An SNMP entity with command responder and/or notification originator applications
- SNMP proxy agent: An SNMP entity with proxy forwarder application
- SNMP mid-level manager: An SNMP entity with command generator and/or notification receiver, plus command responder and/or notification originator applications
- SNMP network management stations: entities with all SNMP applications and possibly other types of applications for managing a very large number of managed nodes

SNMP Manager



SNMPv3 Agent


Network
UDP
SNMP Entity
Dispatcher
Transport Mapping
Message Dispatcher
PDU Dispatcher
Message Processing Subsystem
V1 MP
V2c MP
V3 MP
other MP
Security Subsystem
User-based Security Model
Other Security Model
Access Control Subsystem
View-based Access Control Model
Other Access Control Model
SNMP Engine
Command Responder Application
Notification Originator Application
Notification Receiver Application
SNMP Applications

Section 5
SNMPv3 Textual Conventions

SNMPv3 – Textual Conventions

- SnmpEngineID
  - SNMP engine’s administratively assigned unique identifier.
    - Used for identification and not for addressing.
    - Resolves to an OCTET STRING between 5 and 32 bytes long
  - The very first bit is used to indicate how the rest of the data is composed:
    - ‘0’ - as defined by enterprise
    - ‘1’ - as defined by standard architecture.

SNMPv3 – Textual Conventions

- SnmpEngineID Standard Definition
  - The length of the octet string varies
  - The very first bit is set to 1
  - The first four octets are used for the private enterprise number.
  - The fifth octet indicates how the rest (6th and following octets) are formatted:

    0 - reserved, unused.

    1 - IPv4 address (4 octets)

    2 - IPv6 address (16 octets)

    3 - MAC address (6 octets)

    4 - Text, administratively assigned (maximum length 27)

    5 - Octets, administratively assigned (maximum length 27)

    6-127 - reserved, unused

    128-255 - as defined by the enterprise (maximum length 27)

SNMPv3 – Textual Conventions

- SnmpEngineID Enterprise Definition
  - Has a fixed length of 12 octets
  - The very first bit is set to 0
  - The first four octets are set to the private enterprise number
  - The remaining eight octets are determined via one or more enterprise-specific methods. For example, it may be the IP address of the SNMP entity, or the MAC address of one of the interfaces.

SNMPv3 – Textual Conventions

- snmpSecurityModel
  - Uniquely identifies a Security Model of the Security Subsystem
  - Resolves to an INTEGER
    - ‘0’, reserved for any.
    - ‘1’, SNMPv1
    - ‘2’, SNMPv2c
    - ‘3’, User-Based Security Model (USM)
    - 4-255 reserved for future use
    - Values greater than 255 are allocated to enterprise-specific Security Models as follows:

      enterpriseID * 256 + securityModel
    - This allows enterprises to define up to 255 enterprise specific security models and up to 2^{23}-1 (8,388,606) enterprises

SNMPv3 – Textual Conventions

- snmpSecurityLevel
  - Resolves to an INTEGER
  - Defines the 3 security levels that can be used

    noAuthNoPriv (1)

    SNMP messages are sent without authentication and without privacy

    authNoPriv (2)

    SNMP messages are sent with authentication but without privacy

    authPriv (3)

    SNMP messages are sent with authentication and with privacy

SNMPv3 – Textual Conventions

- SnmpMessageProcessingModel
  - Uniquely identifies the message processing model
  - Resolves to an INTEGER:
    - ‘0’, SNMPv1
    - ‘1’, SNMPv2c
    - ‘2’, SNMPv2u or SNMPv2*
    - ‘3’, SNMPv3
    - 4-255, reserved for standards-track message processing models.
    - Values greater than 255 are handled the same way as with the ‘snmpSecurityModel’:

      enterpriseID * 256 + messageProcessingModel
    - This allows enterprises to define up to 255 message processing models.

SNMPv3 – Textual Conventions

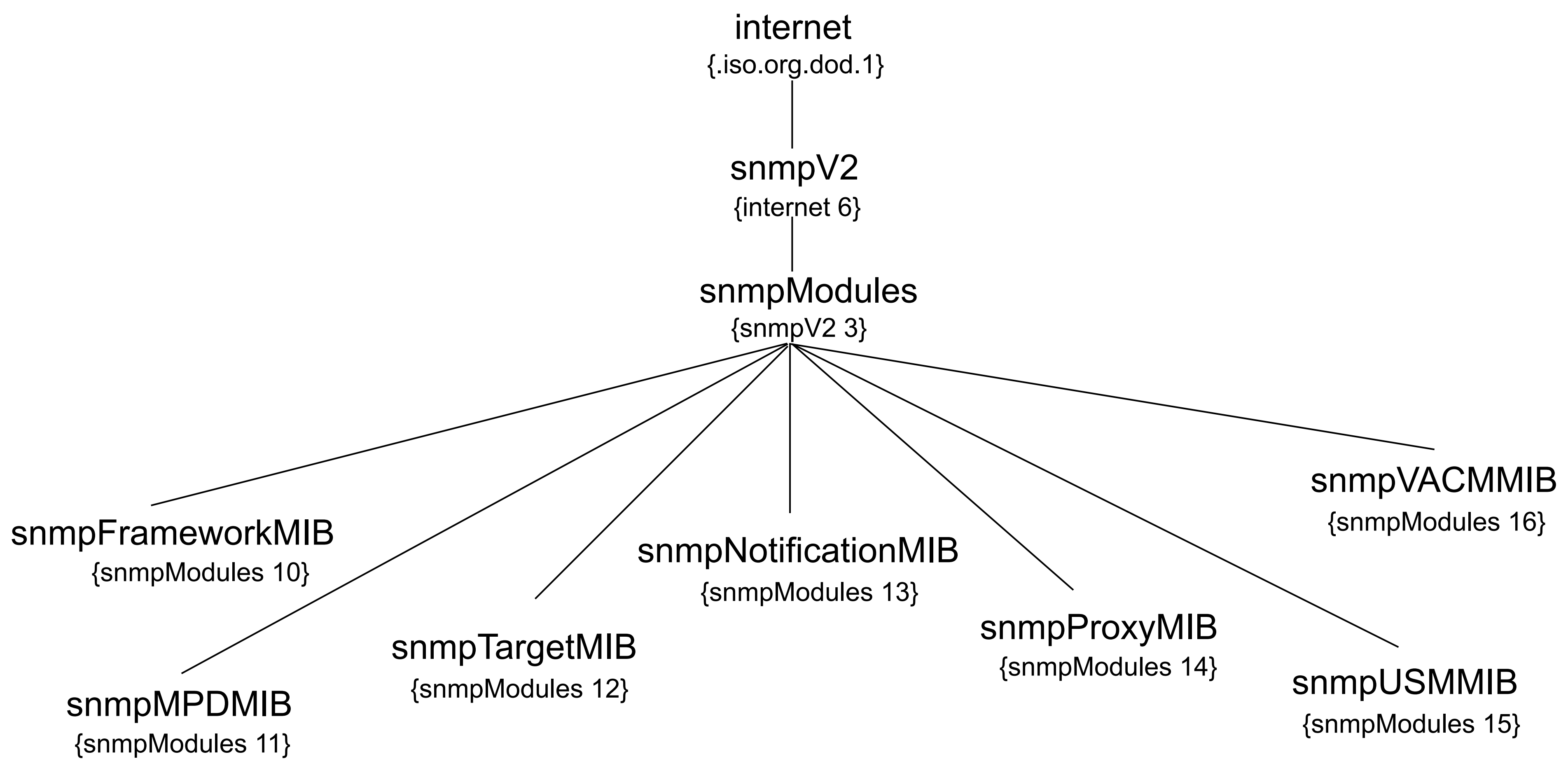
- snmpAdminString
  - Resolves to an OCTET STRING (Up to 255 bytes long)
  - Used to represent administrative information in human readable form.
  - Represents textual information such as a user name or an identifier string

Section 6
SNMPv3 MIBs

SNMP Local Configuration Datastore

- The SNMP entity retains its own sets of configuration information.
- Part of the configuration information is accessible via managed objects and is published in SNMPv3 MIBs.

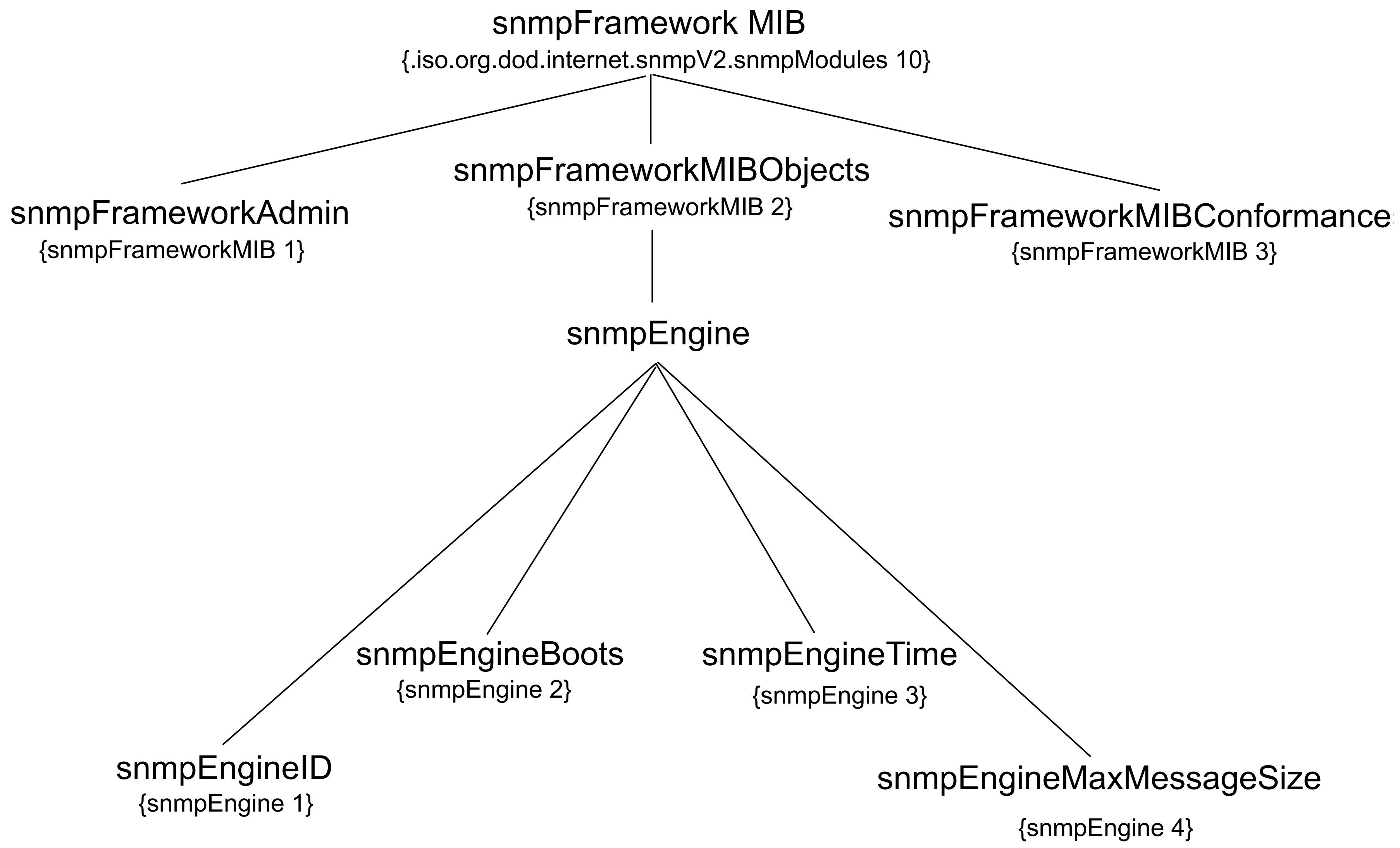
SNMPv3 MIBs


internet
{.iso.org.dod.1}
snmpV2
{internet 6}
snmpModules
{snmpV2 3}
snmpFrameworkMIB
{snmpModules 10}
snmpMPDMIB
{snmpModules 11}
snmpTargetMIB
{snmpModules 12}
snmpNotificationMIB
{snmpModules 13}
snmpProxyMIB
{snmpModules 14}
snmpUSMMIB
{snmpModules 15}
snmpVACMMIB
{snmpModules 16}

SNMPv3 MIBs – Entity

- SNMP-FRAMEWORK-MIB
  - RFC 2571, “An Architecture for Describing SNMP Management Frameworks”
  - {internet snmpV2 snmpModules 10}
- SNMP-MPD-MIB
  - RFC 2572, “Message Processing and Dispatching for the Simple Network Management protocol”
  - {internet snmpV2 snmpModules 11}

SNMPv3 – Framework MIB



Framework MIB

snmpEngineID OBJECT-TYPE

SYNTAX SnmpEngineID

MAX-ACCESS read-only

STATUS current

DESCRIPTION "An SNMP engine's administratively-unique identifier. This information SHOULD be stored in non-volatile storage so that it remains constant across re-initializations of the SNMP engine. "

::= { snmpEngine 1 }

snmpEngineBoots OBJECT-TYPE

SYNTAX INTEGER (1..2147483647)

MAX-ACCESS read-only

STATUS current

DESCRIPTION "The number of times that the SNMP engine has initialized itself since snmpEngineID was last configured. "

::= { snmpEngine 2 }

Framework MIB

snmpEngineTime OBJECT-TYPE

SYNTAX INTEGER (0..2147483647)

UNITS "seconds"

MAX-ACCESS read-only

STATUS current

DESCRIPTION "The number of seconds since the value of the snmpEngineBoots object last changed. When incrementing this object's value would cause it to exceed its maximum, snmpEngineBoots is incremented as if a re-initialization had occurred, and this object's value consequently reverts to zero. "

::= { snmpEngine 3 }

snmpEngineMaxMessageSize OBJECT-TYPE

SYNTAX INTEGER (484..2147483647)

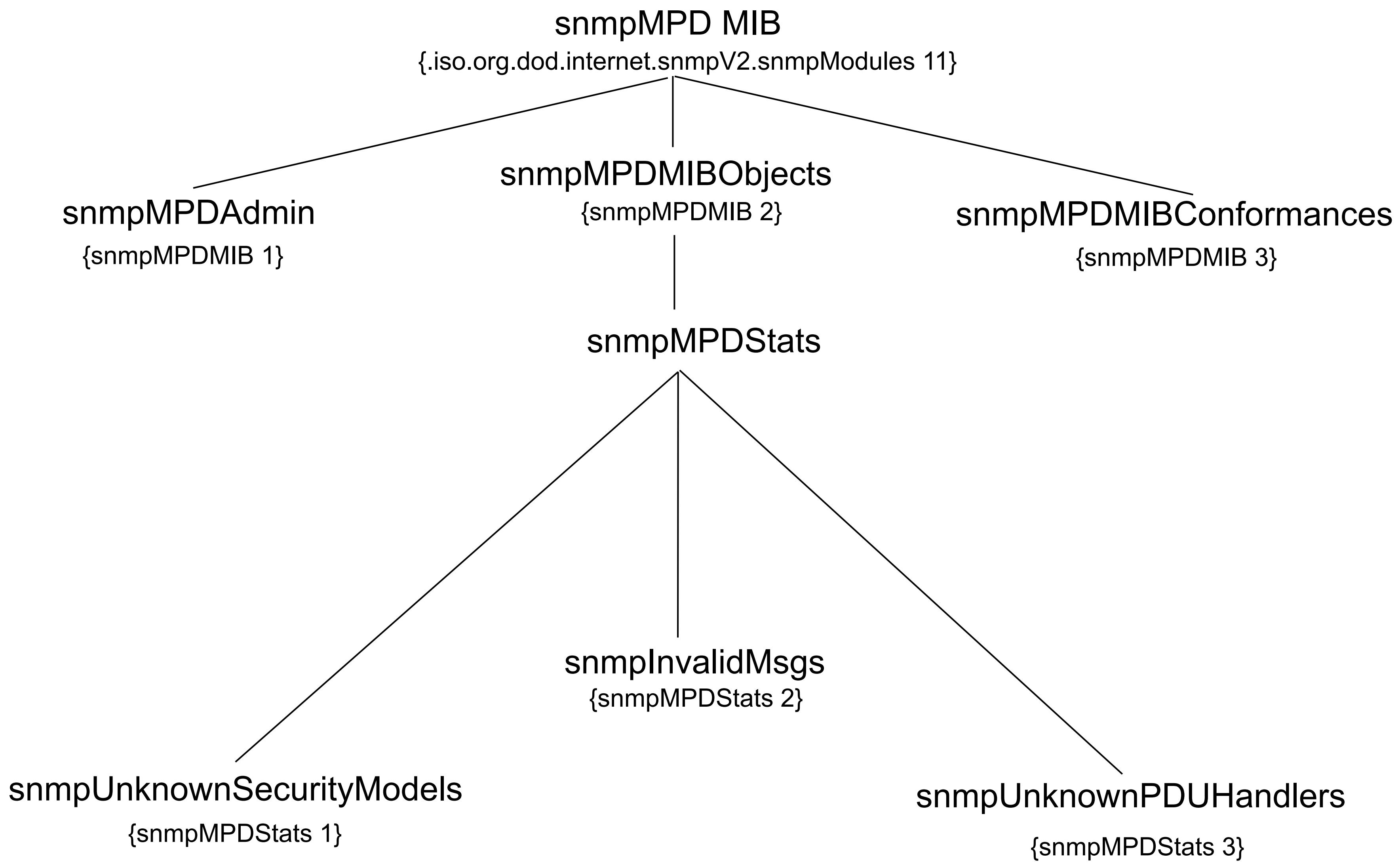
MAX-ACCESS read-only

STATUS current

DESCRIPTION "The maximum length in octets of an SNMP message which this SNMP engine can send or receive and process, determined as the minimum of the maximum message size values supported among all of the transports available and supported by the engine. "

::= { snmpEngine 4 }

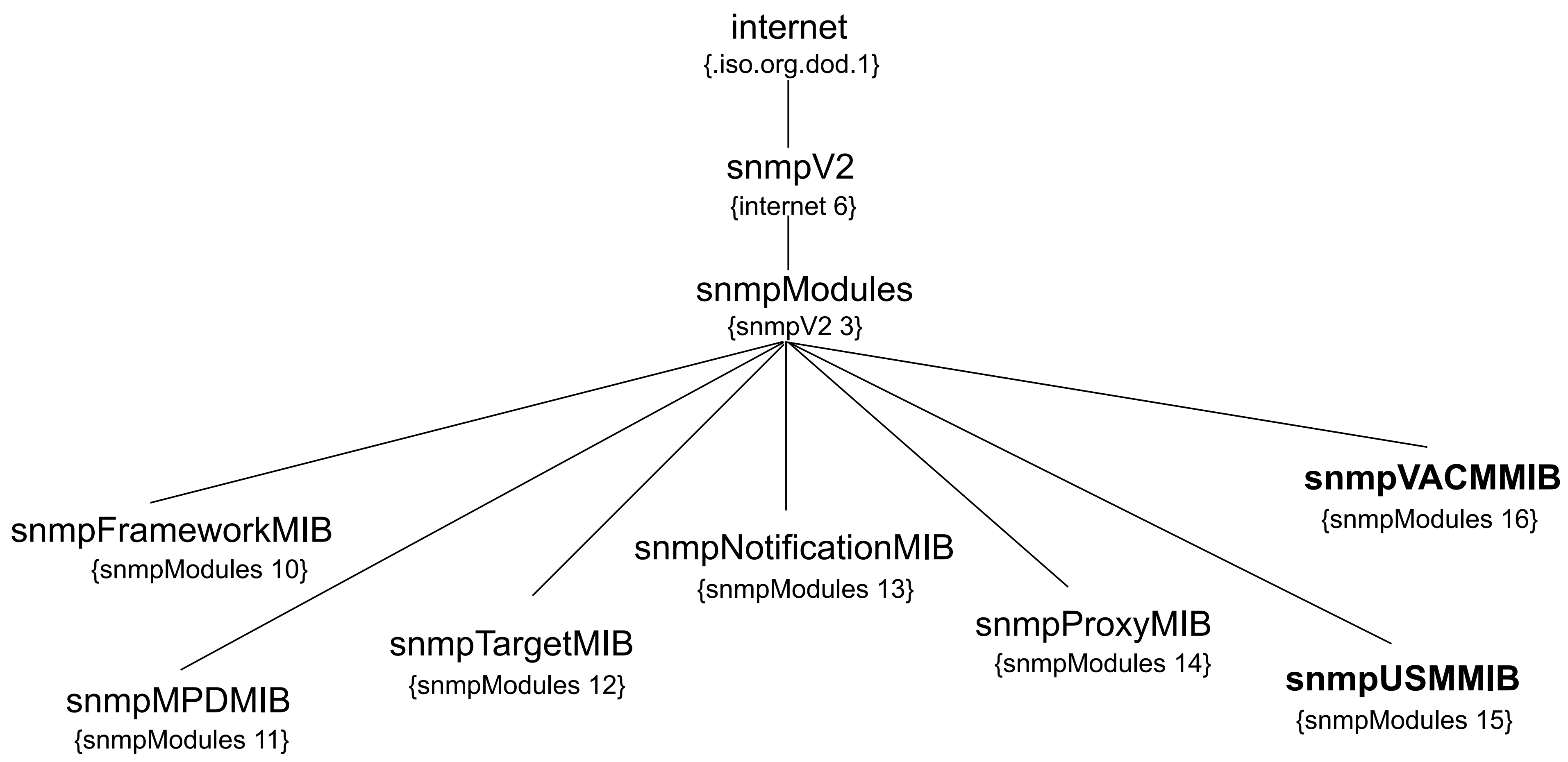
SNMPv3 – MPD MIB



SNMPv3 – MPD Statistics

```
snmpUnknownSecurityModels OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The total number of packets received by the SNMP engine which were
          dropped because they referenced a securityModel that was not known to or
          supported by the SNMP engine. "
::= { snmpMPDStats 1 }
snmpInvalidMsgs OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The total number of packets received by the SNMP engine which were
          dropped because there were invalid or inconsistent components in the SNMP
          message. "
::= { snmpMPDStats 2 }
snmpUnknownPDUHandlers OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The total number of packets received by the SNMP engine which were
          dropped because the PDU contained in the packet could not be passed to an
          application responsible for handling the pduType, e.g. no SNMP application had
          registered for the proper combination of the contextEngineID and the pduType. "
::= { snmpMPDStats 3 }
```

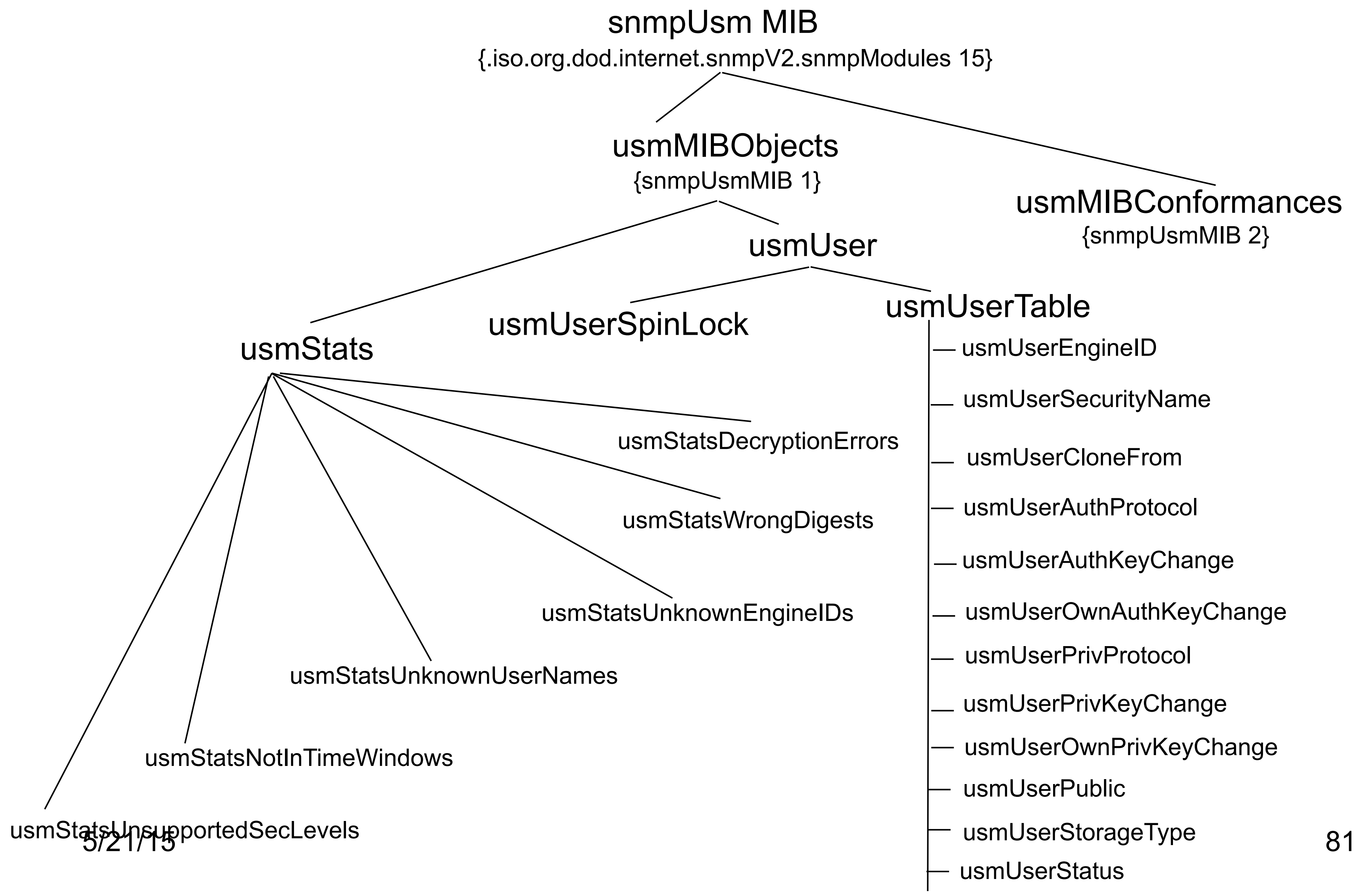
SNMPv3 Authentication & Privacy MIBs



SNMPv3 MIBs – Authentication & Privacy

- SNMP-USM-MIB
  - RFC 2574, “User-based Security Model for version 3 of the Simple Network Management Protocol (SNMPv3)”
  - {snmpModules 15}
- SNMP-VACM-MIB
  - RFC 2575, “View-based Access Control Model for the Simple Network Management Protocol”
  - {snmpModules 16}

SNMPv3 – USM MIB



SNMPv3 – USM MIB

- Defines a table of valid users for the User-based Security Model (usmUserTable)
- Defines authentication and privacy protocols
- Defines USM Error Statistics
- A TestAndIncr object, ‘usmUserSpinLock’, used to coordinate ‘set’ operations to the usmUserTable

SNMPv3 – USM MIB

- usmUserAuthProtocol values defined so far:

Authentication Protocol	Value	Object Identifier
None	usmNoAuthProtocol	{snmpAuthProtocols 1}
HMAC-MD5-96	usmHMACMD5AuthProtocol	{snmpAuthProtocols 2}
HMAC-SHA-96	usmHMACSHAAuthProtocol	{snmpAuthProtocols 3}

SNMPv3 – USM MIB

- usmUserPrivProtocol values defined so far:

Authentication Protocol	Value	Object Identifier
None	usmNoPrivProtocol	{snmpPrivProtocols 1}
CBC-DES	usmDESPrivProtocol	{snmpPrivProtocols 2}

SNMPv3 – USM User Table

Object	Type	Access
usmUserEngineID	SnmpEngineID	not-accessible
usmUserName	SnmpAdminString	not-accessible
usmUserSecurityName	SnmpAdminString	read-only
usmUserCloneFrom	RowPointer	read-create
usmUserAuthProtocol	AutonomousType	read-create
usmUserAuthKeyChange	KeyChange	read-create
usmUserOwnAuthKeyChange	KeyChange	read-create
usmUserPrivProtocol	AutonomousType	read-create
usmUserPrivKeyChange	KeyChange	read-create
usmUserOwnPrivKeyChange	KeyChange	read-create
usmUserPublic	OCTET STRING	read-create
usmUserStorageType	StorageType	read-create
usmUserStatus	RowStatus	read-create

SNMPv3 – USM User Table

- ‘usmUserTable’ maintains authentication and privacy information for each user
- Each user is indexed by an SNMP engine identifier and a user name
- The SNMP engine identifier is mostly local but remote engine identifiers can exist
- Each row specifies the authentication protocol and the encryption protocol to use when exchanging SNMP messages for a specific user
- Each row also contains authentication and encryption KeyChange objects to allow secret keys to be changed remotely

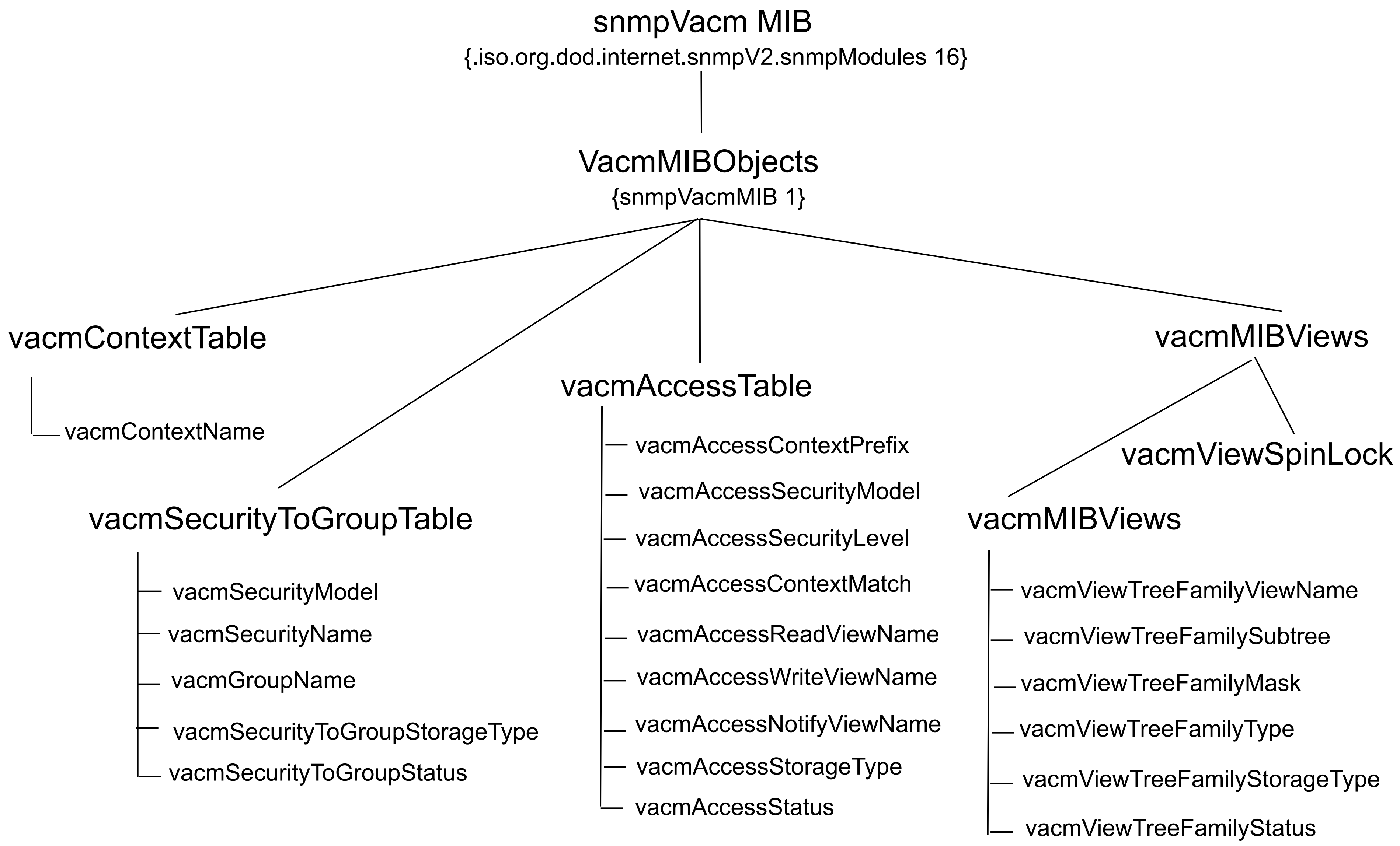
SNMPv3 – USM User Table

- New rows in the usmUserTable are created by cloning them from existing rows
- Atleast one template row must be created and have its secret keys configured through some means other than SNMP (through a command line interface) so that the new rows can be cloned from it.

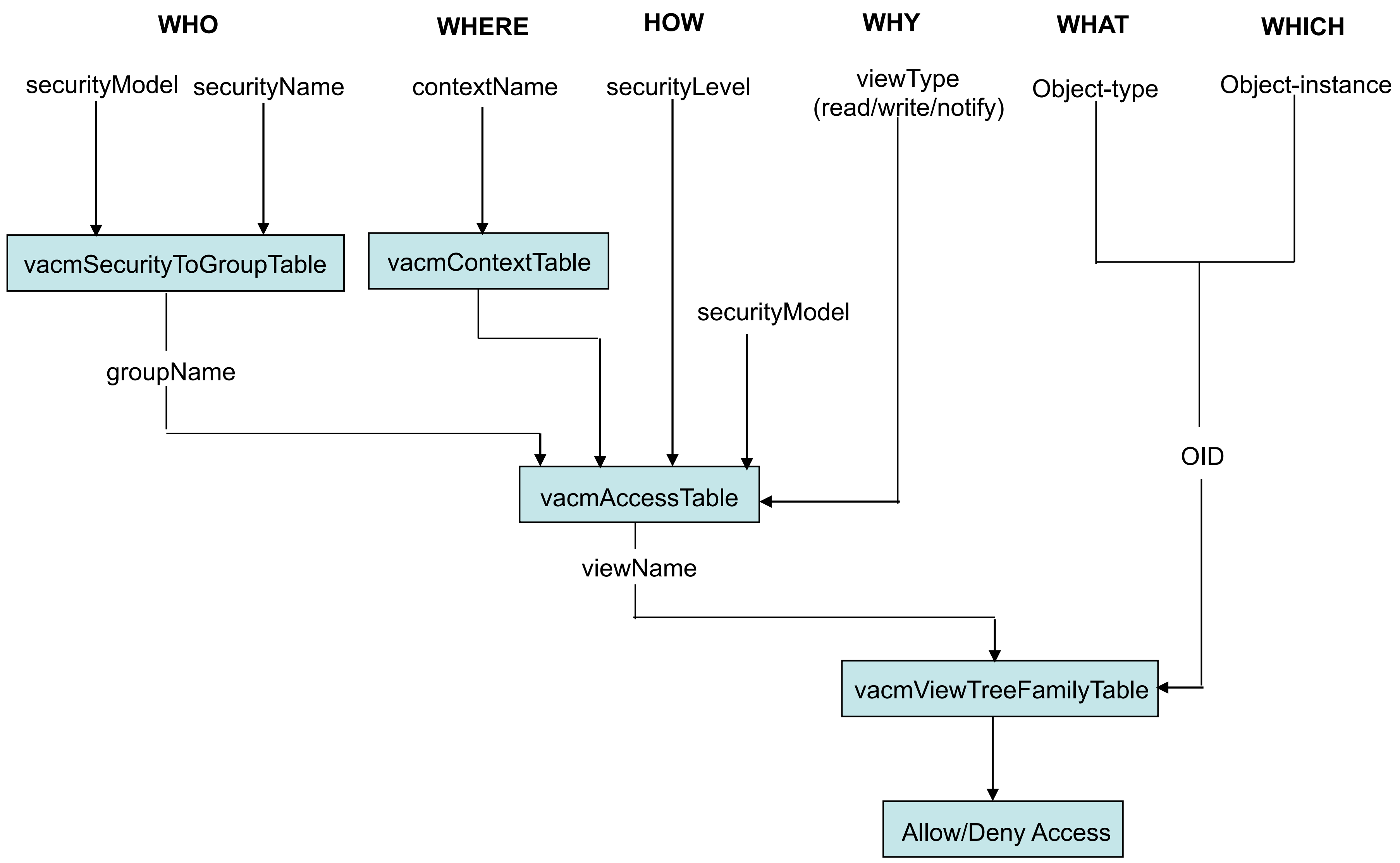
SNMPv3 – USM Error Statistics

- usmStatsUnsupportedSecLevels
  - 32-bit Counter
  - Number of packets dropped because of the requested security level is not supported.
- usmStatsNotInTimeWindows
  - 32-bit Counter
  - Number of packets dropped because they were outside an authoritative engine’s time window
- usmStatsUnknownUserNames
  - 32-bit Counter
  - Number of packets dropped because they referenced an unknown user

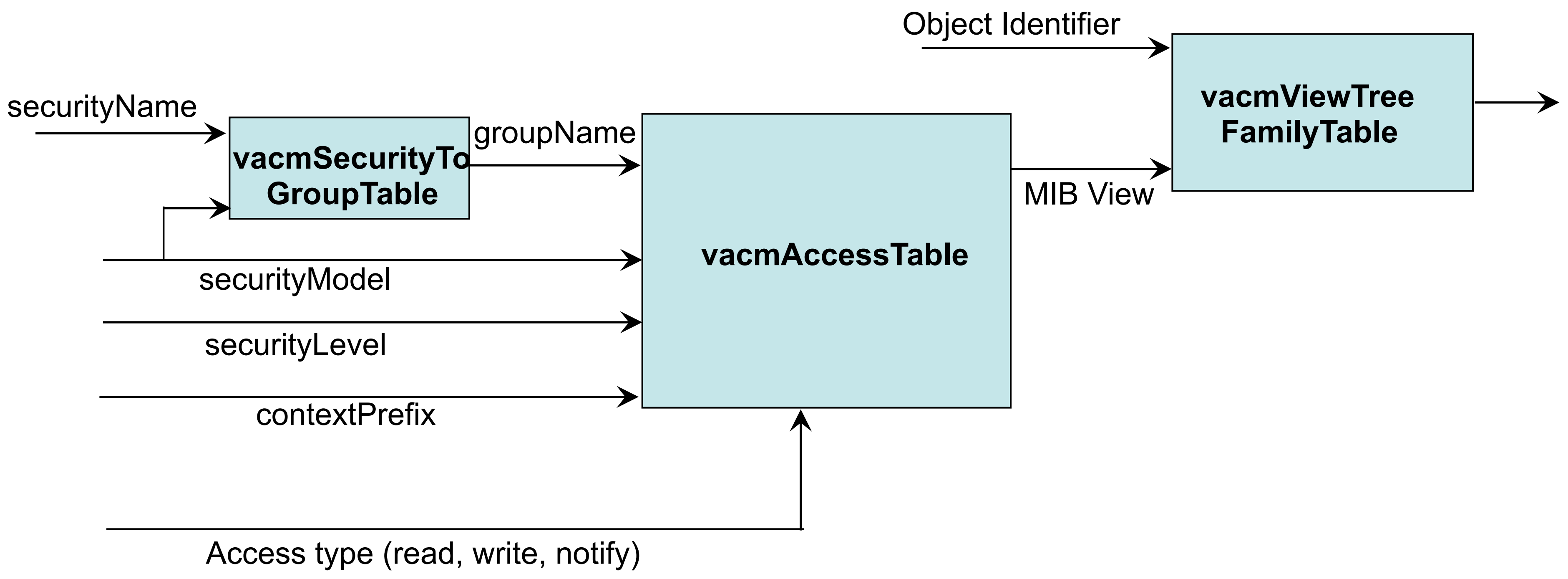
SNMPv3 – VACM MIB



SNMPv3 - VACM MIB



SNMPv3 – VACM Call Flow



VACM MIB

vacmSecurityToGroupTable indexed by vacmSecurityModel.vacmSecurityName

Object	Type	Access
vacmSecurityModel	SnmpSecurityModel	not-accessible
vacmSecurityName	SnmpAdminString	not-accessible
vacmGroupName	SnmpAdminString	read-create
vacmSecurityToGroupStorageType	StorageType	read-create
vacmSecurityToGroupStatus	RowStatus	read-create

VACM MIB

vacmAccessTable indexed by
vacmGroupName.vacmAccessContextPrefix.vacmAccessSecurityModel.vacmAccessSecurityLevel

Object	Type	Access
vacmAccessContextPrefix	SnmpAdminString	not-accessible
vacmAccessSecurityModel	SnmpSecurityModel	not-accessible
vacmAccessSecurityLevel	SnmpSecurityLevel	not-accessible
vacmAccessContextMatch	INTEGER	read-create
vacmAccessReadViewName	SnmpAdminString	read-create
vacmAccessWriteViewName	SnmpAdminString	read-create
vacmAccessNotifyViewName	SnmpAdminString	Read-create
vacmAccessStorageType	Storage Type	read-create

VACM MIB

vacmViewTreeFamilyTable indexed by vacmViewTreeFamilyViewName.vacmViewTreeFamilySubtree

Object	Type	Access
vacmViewTreeFamilyViewName	SnmpAdminString	not-accessible
vacmViewTreeFamilySubtree	OBJECT IDENTIFIER	not-accessible
vacmViewTreeFamilyMask	OCTET STRING	not-accessible
vacmViewTreeFamilyType	INTEGER {included (1), excluded (2)}	read-create
vacmViewTreeFamilyStorageType	StorageType	read-create
vacmViewTreeFamilyStatus	RowSatus	read-create

VACM – Status Codes

Status Codes returned by the VACM subsystem

Code	Description	Error Status
accessAllowed	A MIB view was found, access granted	
notInView	A MIB view found, but access denied	Get: ‘noSuchInstance’ exception Set: ‘noAccess’ error status
noSuchView	No MIB view was found in vacmAccessTable	‘authorizationError’ error status
noSuchContext	The contextName was not found in vacmContextTable	No response PDU generated Increment ‘snmpUnknownContexts’
noGroupName	The securityName and securityModel could not be mapped into a groupName	‘authorizationError’ error status
noAccessEntry	A row could not be found in vacmAccessTable	‘authorizationError’ error status
otherError	Some other undefined error occurred	‘genError’ error status

VACM – Example Configuration

Example vacmSecurityToGroupTable Configuration

	3.initial
vacmSecurityModel	3 (USM)
vacmSecurityName	“initial”
vacmGroupName	“initial”
vacmSecurityToGroupStorageType	nonVolatile
vacmSecurityToGroupStatus	active

VACM – Example Configuration

Example vacmAccessTable Configuration

	initial.3.1	initial.3.2	initial.3.3
vacmGroupName	initial	initial	initial
vacmAccessContextPrefix	“”	“”	“”
vacmAccessSecurityModel	3 (USM)	3 (USM)	3 (USM)
vacmAccessSecurityLevel	1 (noAuthNoPriv)	2 (AuthNoPriv)	3 (AuthPriv)
vacmAccessContextMatch	exact	exact	exact
vacmAccessReadViewName	restricted	internet	internet
vacmAccessWriteViewName	“”	internet	internet
vacmAccessNotifyViewName	restricted	internet	internet

VACM – Example Configuration

Example vacmViewTreeFamily for mimimum security

	internet.1.3.6.1	restricted.1.3.6.1
vacmViewTreeFamilyViewName	internet	restricted
vacmViewTreeFamilySubtree	1.3.6.1	1.3.6.1
vacmViewTreeFamilyMask	“”	“”
vacmViewTreeFamilyType	1 (included)	1 (included)
vacmViewTreeFamilyStorageType	nonVolatile	nonVolatile
vacmViewTreeFamilyStatus	active	active

VACM – Example Configuration

Example vacmViewTreeFamily for semi-secure configuration

	internet.1.3.6.1	restricted.1.3.6.1.2.1.1
vacmViewTreeFamilyViewName	internet	restricted
vacmViewTreeFamilySubtree	1.3.6.1	1.3.6.1.2.1.1
vacmViewTreeFamilyMask	“”	“”
vacmViewTreeFamilyType	1 (included)	1 (included)
vacmViewTreeFamilyStorageType	nonVolatile	nonVolatile
vacmViewTreeFamilyStatus	active	active