Chapter 7

SNMP Management: SNMPv3

Objectives

- SNMPv3 features
  - Documentation architecture
  - Formalized SNMP architecture
  - Security
- SNMP engine ID and name for network entity
- SNMPv3 applications and primitives
- SNMP architecture
  - Integrates the three SNMP versions
  - Message processing module
  - Dispatcher module
  - Future enhancement capability
- User security model, USM
  - Derived from user ID and password
  - Authentication
  - Privacy
  - Message timeliness
- View-based access control model, VACM
  - Configure set of MIB views for agent with contexts
  - Family of subtrees in MIB views
  - VACM process

Key Features

- Modularization of document

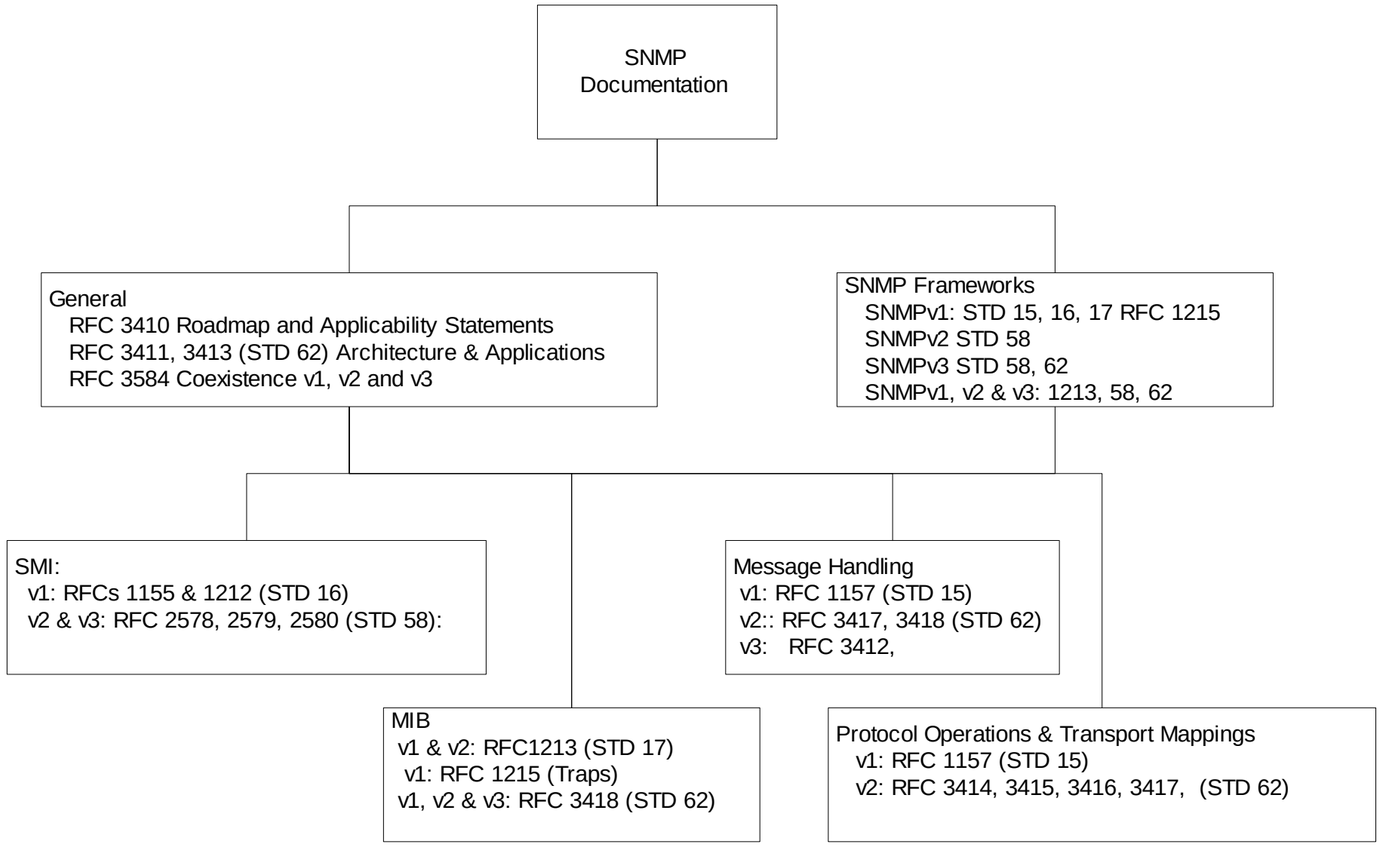
- Modularization of architecture

  – The design of the architecture integrated SNMPv1 and SNMPv2 specifications with the newly proposed SNMPv3. This enables continued usage of legacy SNMP entities along with SNMPv3 agents and manager.

- SNMP engine : An SNMP engine is defined with explicit subsystems that include dispatch and message-processing functions. It manages all three versions of SNMP to coexist in a management entity.

-

Documentation



– Compare this to the document organization in Chapter 4

Table 7.1 SNMPv3 RFCs

RFC 3410	Introduction and Applicability Statements (not STD)
RFC 3411	Architecture for Describing SNMP Management Frameworks
RFC 3412	Message Processing and Dispatching for SNMP
RFC 3413	SNMPv3 Applications
RFC 3414	User-based Security Model (USM) for SNMPv3
RFC 3415	View-based Access Control Model for SNMP
RFC 3416	Version 2 of the Protocol Operations for SNMP
RFC 3417	Transport Mappings for SNMP
RFC 3418	MIB for SNMP
RFC 3584	SNMPv3 Coexistence and Transition (BCP 74)

Architecture

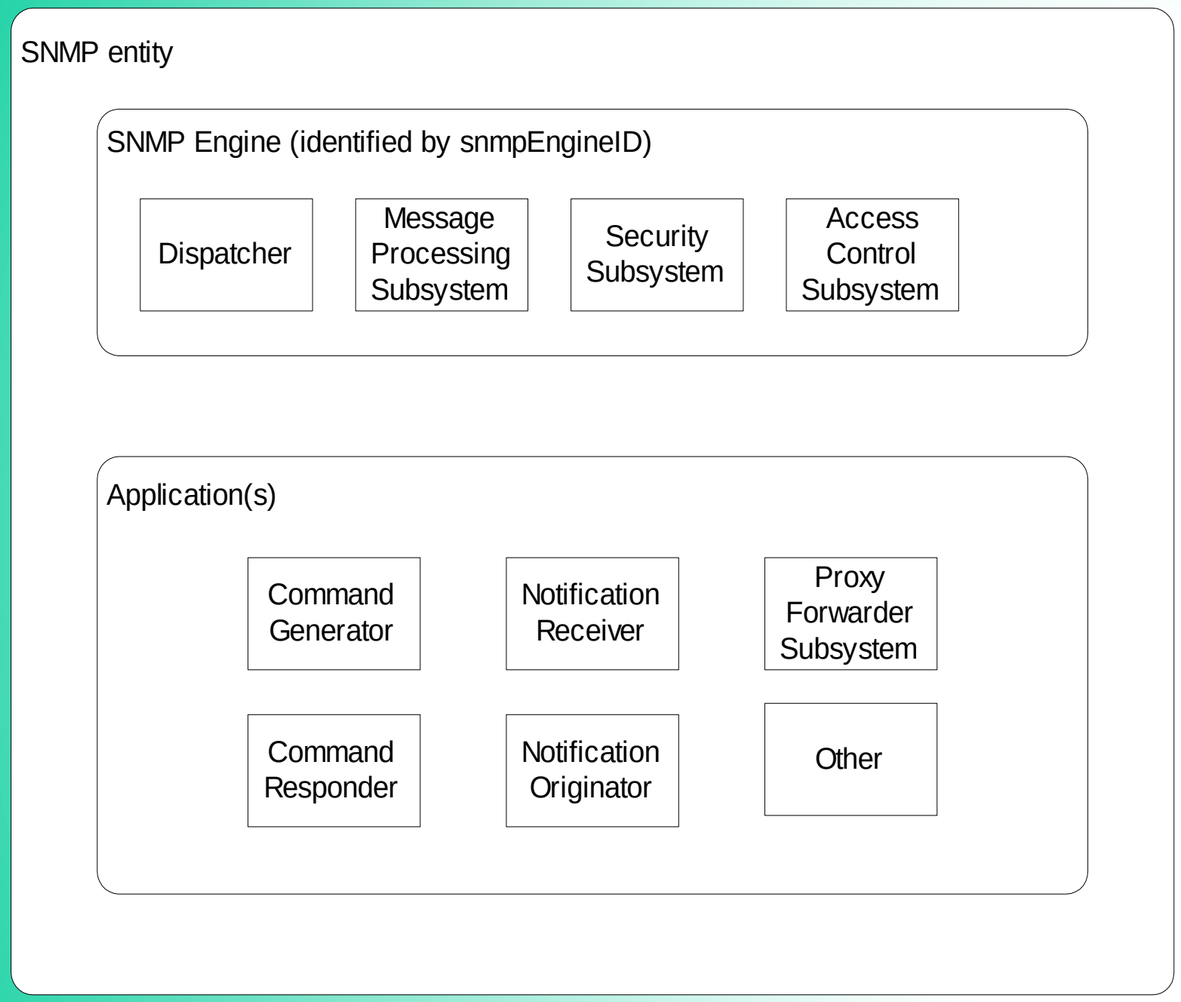


Figure 7.2 SNMPv3 Architecture

Notes

- SNMP entity is a node with an SNMP management element - either an agent or manager or both

- Three names associated with an entity
  - Entities: SNMP engine
  - Identities: Principal and security names
  - Management Information: Context engine

Dispatch Subsystem. There is only one dispatcher in an SNMP engine and it can handle multiple versions of SNMP messages. It does the following three sets of functions. First, it sends messages to and receives messages from the network. Second, it determines the version of the message and interacts with the corresponding MPM. Third, it provides an abstract interface (described in Section 7.3.3) to SNMP applications to deliver an incoming PDU to the local application and to send a PDU from the local application to a remote entity.

7.3.2 Names

Naming of entities, identities, and management information is part of SNMPv3 specifications. We already mentioned the naming of an entity by its SNMP engine ID, *snmpEngineID*. Two names are associated with identities, *principal* and *securityName*. *Principal* is the "who" requesting services. It could be a person or an application. The *securityName* is a human readable string representing a principal. The principal could be a single user; for example, name a network manager or a group of users, such as names of operators in the network operations center. It is made non-accessible. It is hidden and is based on the security model (SM) used. However, it is administratively given a security name; for example, User 1 or Admin, which is made readable by all.

Context

A management entity can be responsible for more than one managed object. For example, a management agent associated with a managed object at a given node could be managing a neighboring node besides its own. Each object is termed *context* and has a *contextEngineID* and a *contextName*.

Key Features

- Security feature :
  - Secure communication (of information): The configuration can be set remotely with secured communication that protects against modification of information and masquerade by using **encryption** schemes. It also tries to ensure against malicious modification of messages by reordering and time delaying of message streams, as well as protects against eavesdropping of messages.

  - Access control: The access policy used in SNMPv1 and SNMPv2 is continued and formalized in the access control in SNMPv3, designated VACM (View-based Access Control Model). The SNMP engine defined in the architecture checks whether a specific type of access (read, write, create, notify) to a particular object (instance) is allowed.

SNMP Engine ID

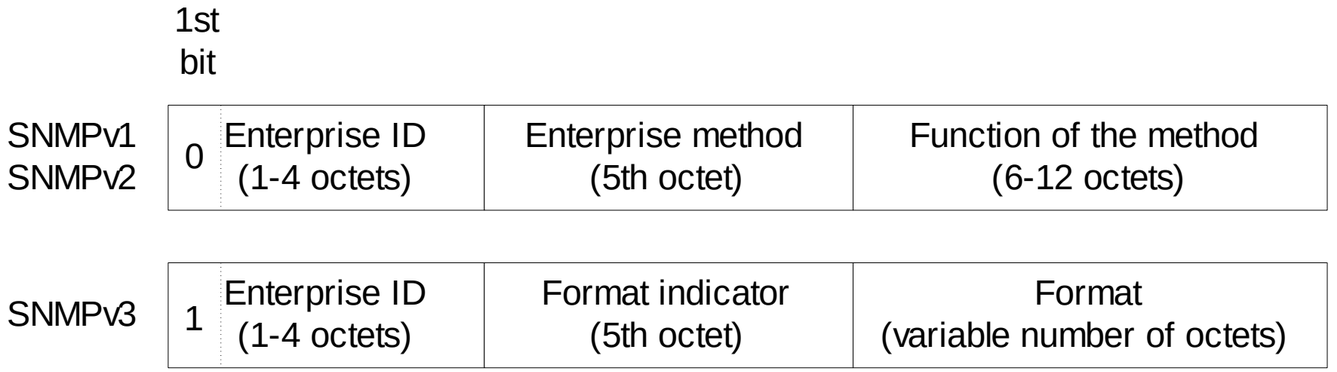


Figure 7.3 SNMP Engine ID

Notes

- Each SNMP engine has a unique ID: *snmpEngineID*
- Acme Networks {enterprises 696}
- SNMPv1 snmpEngineID '000002b8'H
- SNMPv3 snmpEngineID '800002b8'H
  (the 1st octet is 1000 0000)

SNMPv3 Engine ID Format 5th Octet

Notes

- For SNMPv1 and SNMPv2:
  - Octet 5 is the method
  - Octet 6-12 is IP address
- Examples: IBM host IP address 10.10.10.10

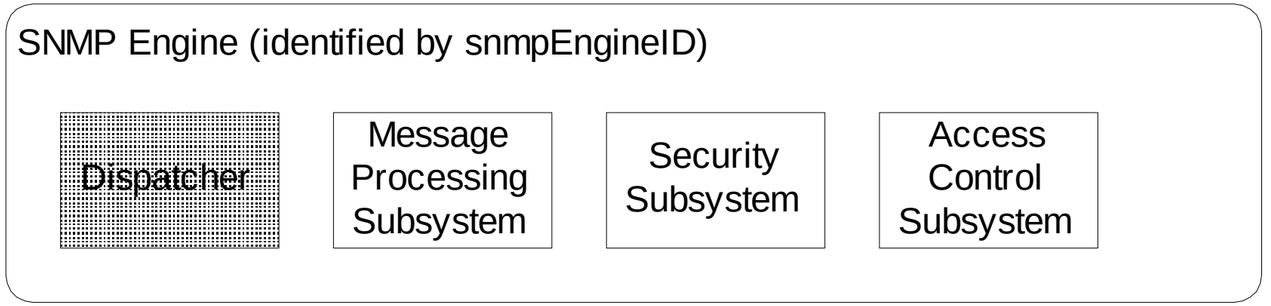
  SNMPv1: 00 00 00 02 01 0A 0A 0A 0A 00 00 00

  SNMPv3: 10 00 00 02 02 00 00 ... 00 00 00 0A 0A 0A 0A

Table 7.2 SNMPv3 Engine ID Format (5th octet)

0	Reserved, unused
1	IPv4 address (4 octets)
2	IPv6 (16 octets) Lowest non-special IP address
3	MAC address (6 octets) Lowest IEEE MAC address, canonical order
4	Text, administratively assigned Maximum remaining length 27
5	Octets, administratively assigned Maximum remaining length 27
6-127	Reserved, unused
128-255	As defined by the enterprises Maximum remaining length 27

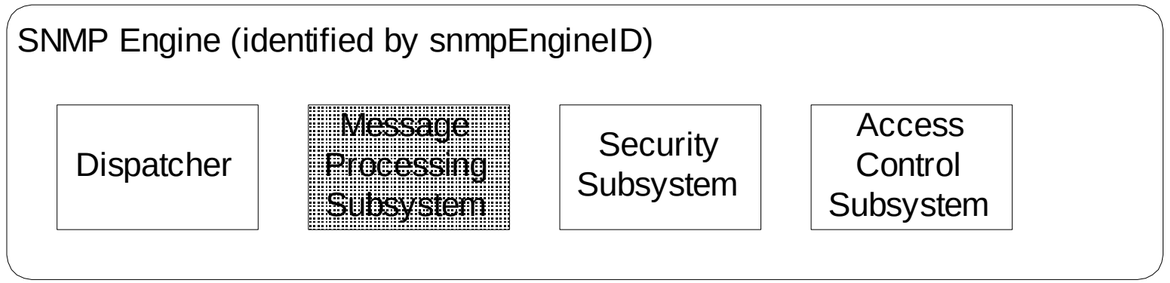
Dispatcher



Notes

- One dispatcher in an SNMP engine
- Handles multiple version messages
- Interfaces with application modules, network, and message processing models
- Three components for three functions
  - Transport mapper delivers messages over the transport protocol
  - Message Dispatcher routes messages between network and appropriate module of MPS
  - PDU dispatcher handles messages between application and MPS

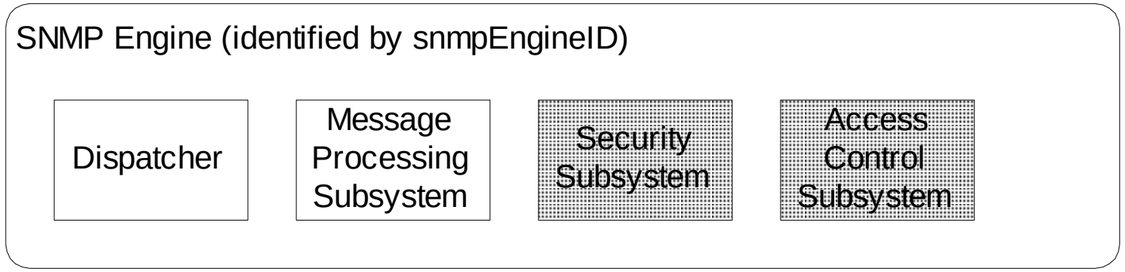
Message Processing Subsystem



Notes

- Contains one or more Message Processing Models
- One MPM for each SNMP version
- SNMP version identified in the header

Security and Access Control



Notes

- Security at the message level
  - Authentication
  - Privacy of message via secure communication
- Flexible access control
  - Who can access
  - What can be accessed
  - Flexible MIB views

Names

- SNMP Engine ID snmpEngineID
- Principal principal
  Who: person or group or application
- Security Name securityName
  human readable name
- Context Engine ID contextEngineID
- Context Name contextName

Notes

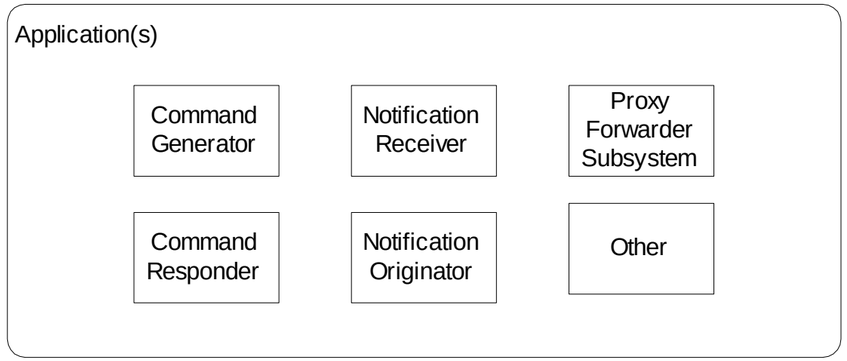
- An SNMP agent can monitor more than one network element (context)

Examples:
SNMP Engine ID IP address

Principal John Smith
Security Name Administrator

Principal Li, David, Kristen, Rashmi,
Security Name Operator

Applications



Notes

Application	Example
• Command generator	get-request
• Command responder	get-response
• Notification originator	trap generation
• Notification receiver	trap processing
• Proxy Forwarder (SNMP versions only)	get-bulk to get-next
• Other	Special application

Abstract Service Interface

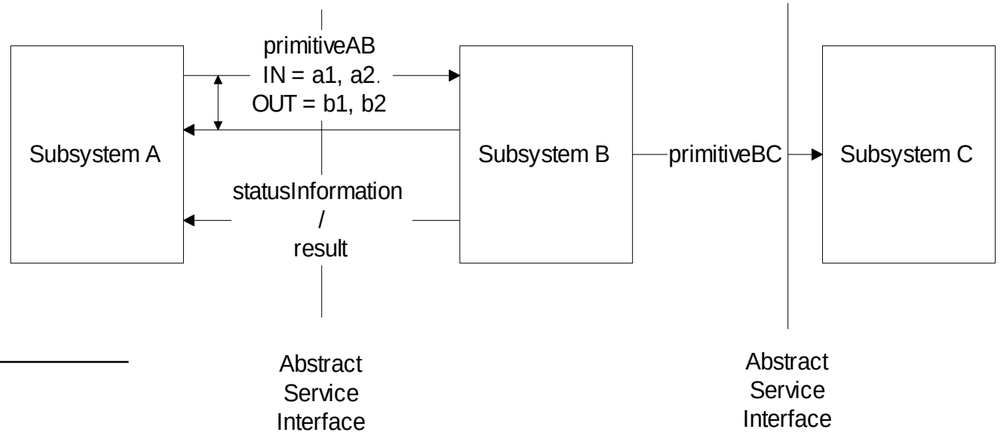


Figure 7.4(a) Abstract Service Interface

Notes

- Abstract service interface is a conceptual interface between modules, independent of implementation
- Defines a set of primitives
- Primitives associated with receiving entities except for Dispatcher
- Dispatcher primitives associated with
  - messages to and from applications
  - registering and un-registering of application modules
  - transmitting to and receiving messages from network
- IN and OUT parameters
- Status information / result

sendPDU Primitive

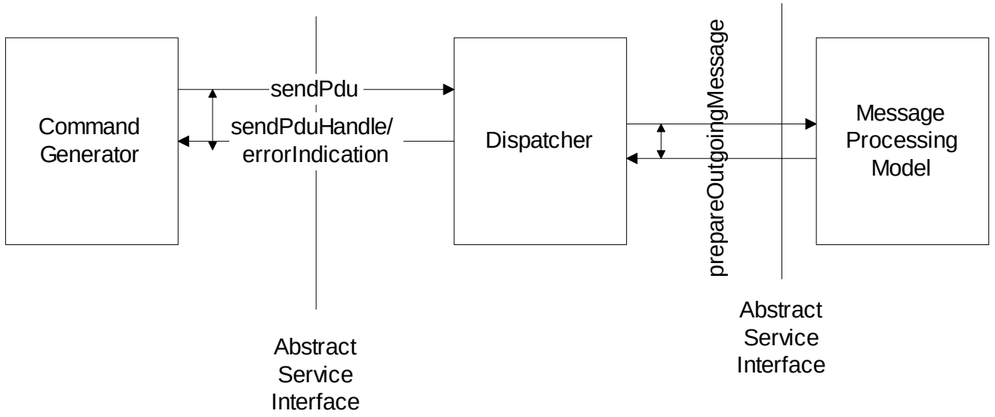


Figure 7.4(b) Abstract Service Interface for sendPdu

Notes

- sendPdu request sent by the application module, command generator, is associated with the receiving module, dispatcher
- After the message is transmitted over the network, dispatcher sends a handle to the command generator for tracking the response
- sendPdu is the IN parameter
- sendPduHandle is the OUT parameter, shown as coupled to the IN parameter

Dispatcher Primitives

Module	Primitive	Service Provided
Dispatcher	sendPdu	Request from application to send a PDU to a remote entity
Dispatcher	processPdu	Processing of incoming message from remote entity
Dispatcher	returnResponsePdu	Request from application to send a response PDU
Dispatcher	processResponsePdu	Processing of incoming response from a remote entity
Dispatcher	registerContextEngineID	Register request from a Context Engine
Dispatcher	unregisterContextEngineID	Unregister request from a Context Engine

Notes

Command Generator

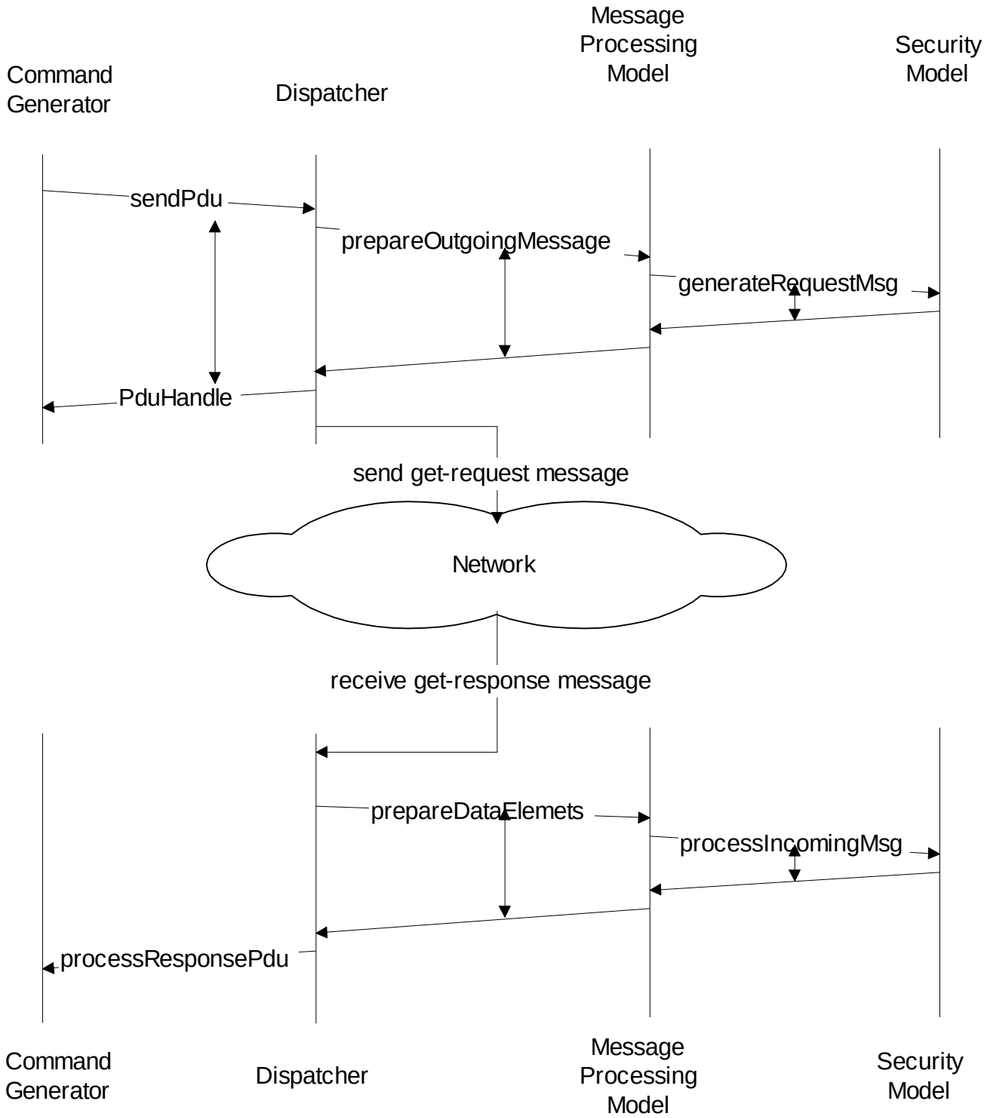


Figure 7.5 Command Generator Application

Command Responder

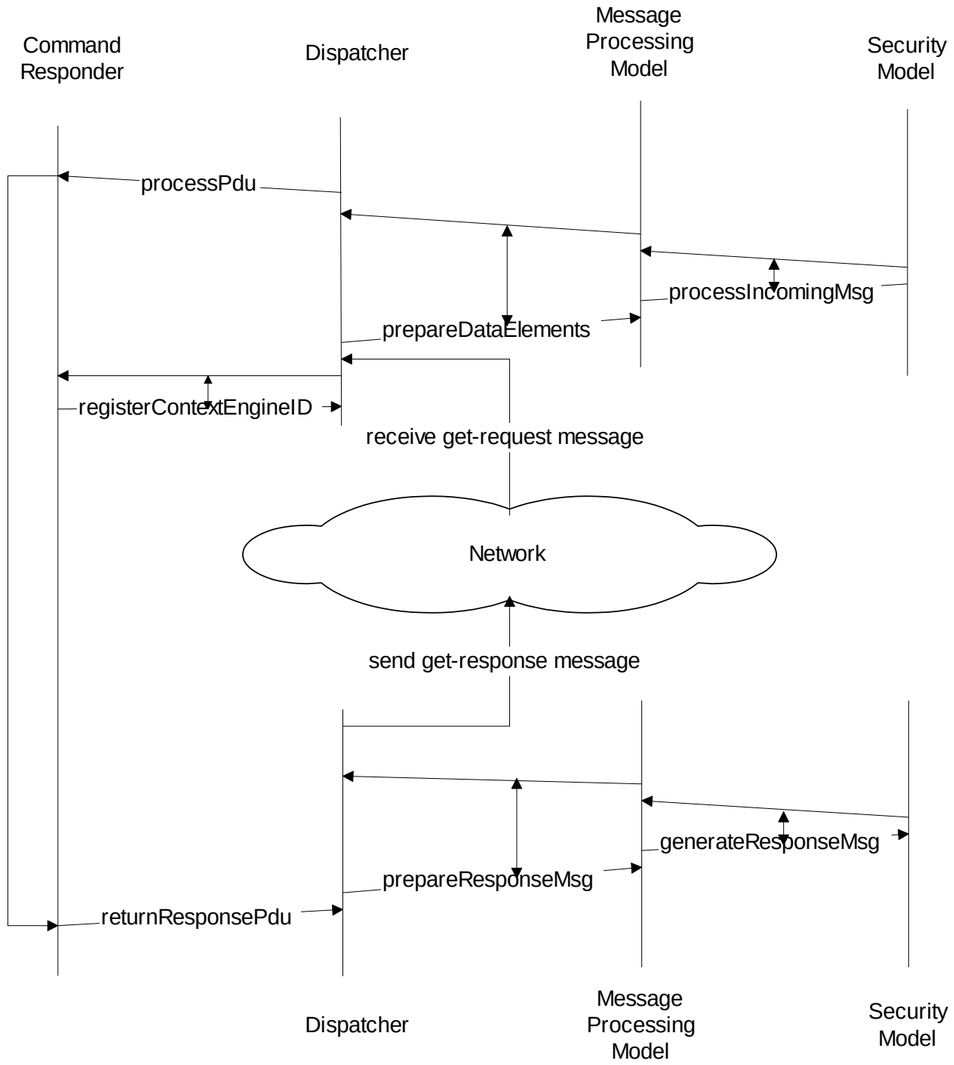


Figure 7.6 Command Responder Application

Notification / Proxy

- Notification originator
  - Generates trap and inform messages
  - Determines target, SNMP version, and security
  - Decides context information
- Notification receiver
  - Registers with SNMP engine
  - Receives notification messages
- Proxy forwarder
  - Proxy server
  - Handles only SNMP messages by
    - Command generator
    - Command responder
    - Notification generator
    - Report indicator
  - Uses the translation table in the proxy group MIB

Notes

SNMpV2 MIB

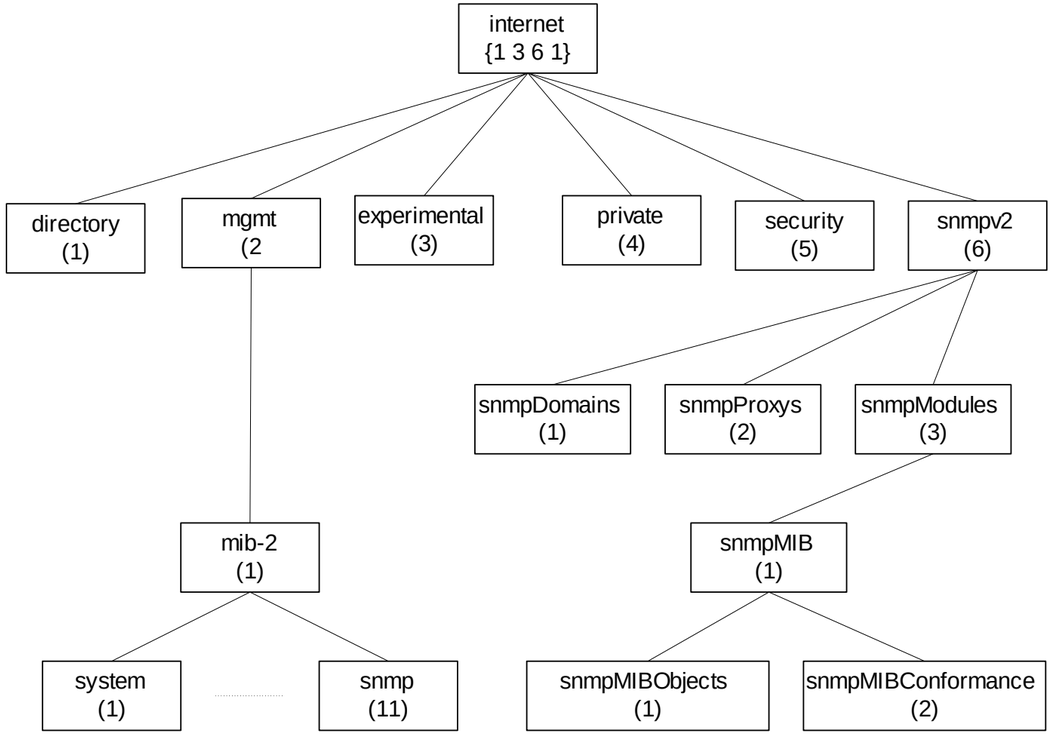


Figure 6.31 SNMPv2 Internet Group

Notes

- **SNMPv3 MIB developed under snmpModules**
- Security placeholder not used

SNMPv3 MIB

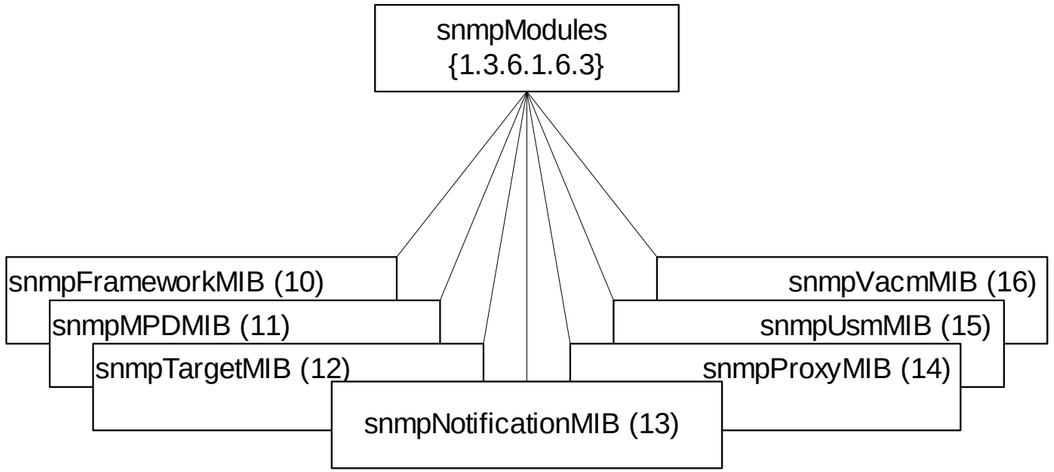


Figure 7.7 SNMPv3 MIB

- snmpFrameworkMIB describes SNMP management architecture
- snmpMPDMIB identifies objects in the message processing and dispatch subsystems
- snmpTargetMIB and snmpNotificationMIB used for notification generation
- snmpProxyMIB defines translation table for proxy forwarding
- snmpUsm MIB defines user-based security model objects
- snmpVacmMIB defines objects for view-based access control

SNMPv3 Target MIB

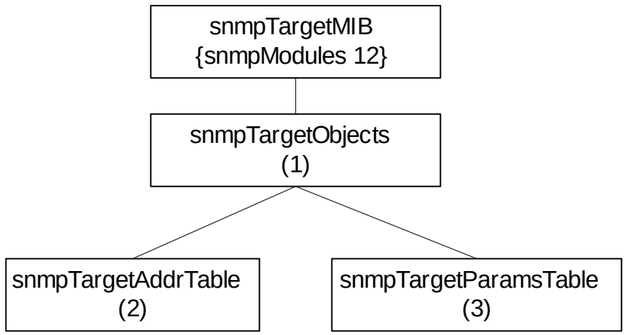


Figure 7.8 Target Address and Target Parameter Tables

Notes

- Target MIB contains two tables
- Target address table contains addresses of the targets for notifications (see notification group)
- Target address table also contains information for establishing the transport parameters
- Target address table contains reference to the second table, target parameter table
- Target parameter table contains security parameters for authentication and privacy

SNMPv3 Notification MIB

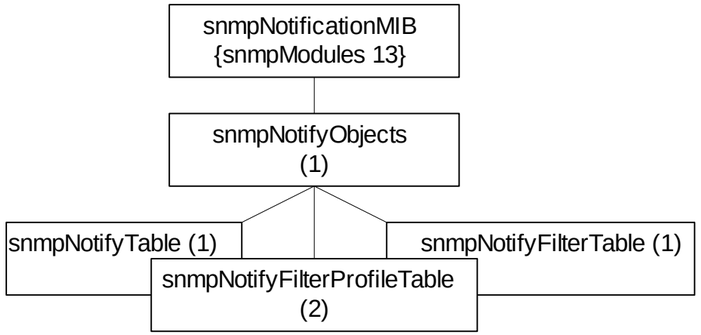


Figure 7.9 SNMP Notification Tables

Notes

- Notification group contains three tables
- Notify table contains groups of management targets to receive notifications and the type of notifications
- The target addresses to receive notifications that are listed in target address table (see target group) are tagged here
- Notification profile table defines filter profiles associated with target parameters
- Notification filter table contains table profiles of the targets

Security

Security Threats

Notes

- Modification of information: Contents modified by unauthorized user, does not include address change
- Masquerade: change of originating address by unauthorized user
- Fragments of message altered by an unauthorized user to modify the meaning of the message
- Disclosure is eavesdropping
- Disclosure does not require interception of message
- Denial of service and traffic analysis are not considered as threats

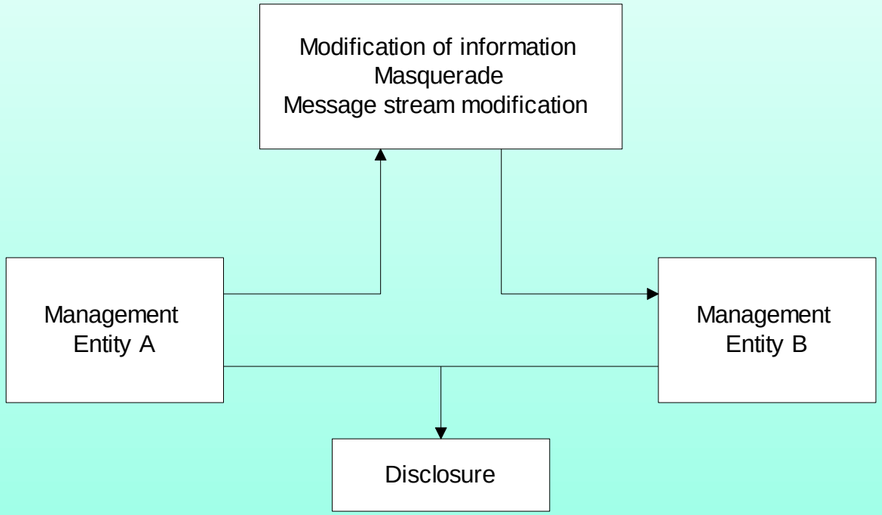


Figure 7.10 Security Threats to Management Information

Security

Security Services

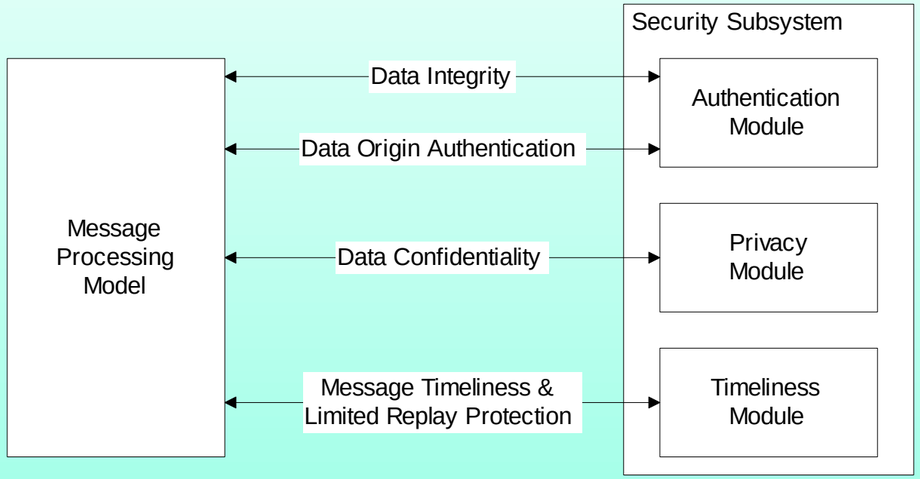
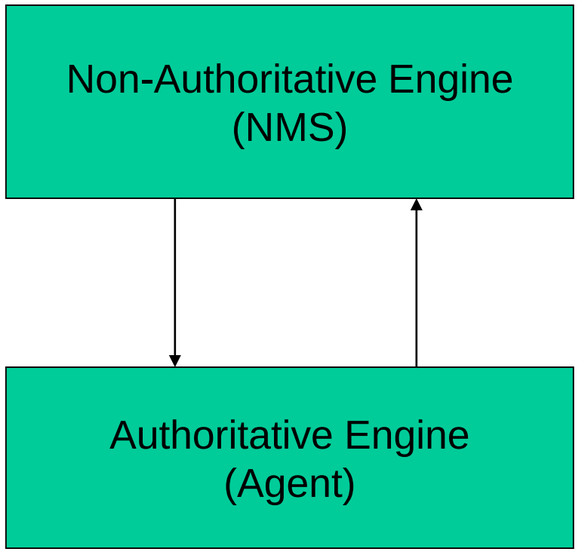


Figure 7.11 Security Services

Notes

- Authentication
  - Data integrity:
    - HMAC-MD5-96 / HMAC-SHA-96
  - Data origin authentication
    - Append to the message a unique Identifier associated with authoritative SNMP engine
- Privacy / confidentiality:
  - Encryption
- Timeliness:
  - Authoritative Engine ID, no. of engine boots and time in seconds

Role of SNMP Engines



Notes

- Responsibility of Authoritative engine:
  - Unique SNMP engine ID
  - Time-stamp
- Non-authoritative engine should keep a table of the time-stamp and authoritative engine ID

SNMP V3 User-Based Security Model

- The User-Based Security Model (USM) uses the concept of an authoritative engine.
- In any message transmission, one of the two entities, transmitter or receiver, is designated as the authoritative SNMP engine.

- **Authorative Engine**
- This is decided based on the following rules:

  - When SNMP message contains a payload that expects a response => the receive of such messages is authoritative.
    - for example, a Get, GetNext, GetBulk, Set, or Inform PDU

  - When an SNMP message contains a payload that does not expect a response => the sender of such a message is authoritative.
    - for example, an SNMPv2-Trap, Response, or Report PDU

SNMP V3 User-Based Security Model

- This designation serves two purposes:

- **Firstly**, the timeliness of a message is determined with respect to a clock maintained by each authoritative engine.
  - Authorative engine sends a message, it contains the current value of its clock
    - Non-authoritative recipient synchronizes on that clock.
    - E.g. authorative sends Trap, Response, Report

- Each non-authorative engine maintains an estimate of the time value for each authorative engine with which it communicates
  - Non-authoritative engine sends a message, it contains current estimate of the time value at the destination
    - destination can assess the timeliness of the message.
    - E.g. Non-authorative sends Get, GetNext, GetBulk, Set, Inform

SNMP V3 User-Based Security Model

- **Secondly,** a key localization process enables a single principal management station to own keys stored in multiple engines.

  - Keys are localized to the authoritative engine in such a way
    - the principal management station is responsible for a single key
    - avoids the security risk of storing multiple copies of the same key in a distributed network.

  - When an outgoing message is passed to the USM by the Message Processor, the USM fills in the security-related parameters in the message header.

  - When an incoming message is passed to the USM by the Message Processor, the USM processes the values contained in those fields.

SNMPv3 Message Format

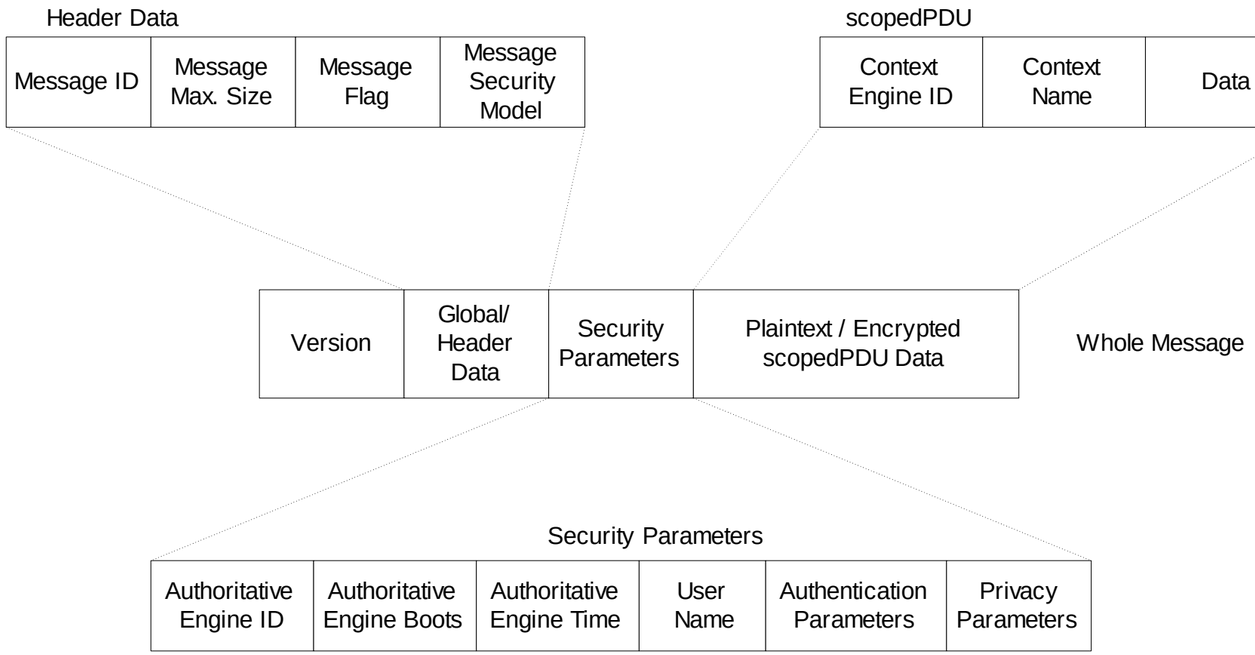


Figure 7.12 SNMPv3 Message Format

Notes

SNMPv3 Message Format

Table 7.7 SNMPv3 Message Format

Field	Object name	Description
Version	msgVersion	SNMP version number of the message format
Message ID	msgID	Administrative ID associated with the message
Message Max. Size	msgMaxSize	Maximum size supported by the sender
Message flags	msgFlags	Bit fields identifying report, authentication, and privacy of the message
Message Security Model	msgSecurityModel	Security model used for the message; concurrent multiple models allowed
Security Parameters (See Table 7.8)	msgSecurityParameters	Security parameters used for communication between sending and receiving security modules
Plaintext/Encrypted scopedPDU Data	scopedPduData	Choice of plaintext or encrypted scopedPDU; scopedPDU uniquely identifies context and PDU
Context Engine ID	contextEngineID	Unique ID of a context (managed entity) with a context name realized by an SNMP entity
Context Name	contextName	Name of the context (managed entity)
PDU	data	Contains unencrypted PDU

User-based Security Model

- Based on traditional user name concept
- USM primitives across abstract service interfaces
  - Authentication service primitives
    - authenticateOutgoingMsg
    - authenticateIncomingMsg
  - Privacy Services
    - encryptData
    - decryptData

Notes

Secure Outgoing Message

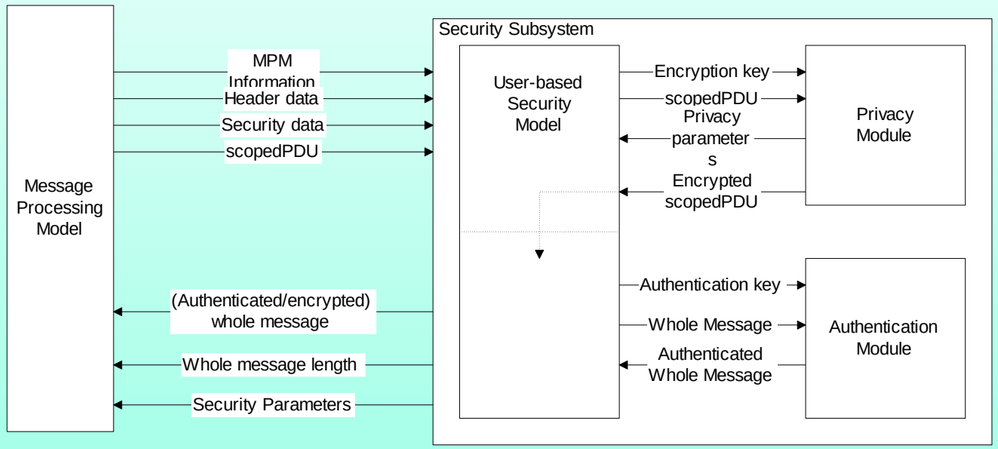


Figure 7.13 Privacy and Authentication Service for Outgoing Message

Notes

- USM invokes privacy module w/ encryption key and scopedPDU
- Privacy module returns privacy parameters and encrypted scopedPDU
- USM then invokes the authentication module with authentication key and whole message and receives authenticated whole message

Secure Incoming Message

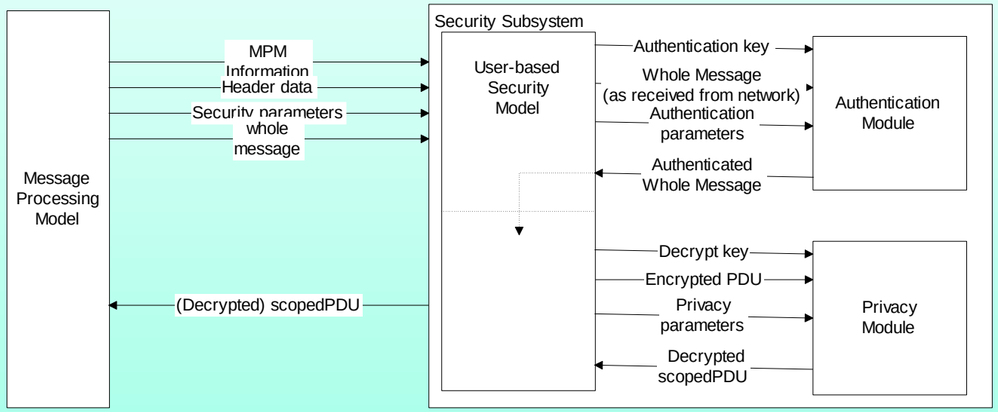


Figure 7.14 Privacy and Authentication Service for Incoming Message

Notes

- Processing secure incoming message reverse of secure outgoing message
- Authentication validation done first by the authentication module
- Decryption of the message then done by the privacy module

Security Parameters

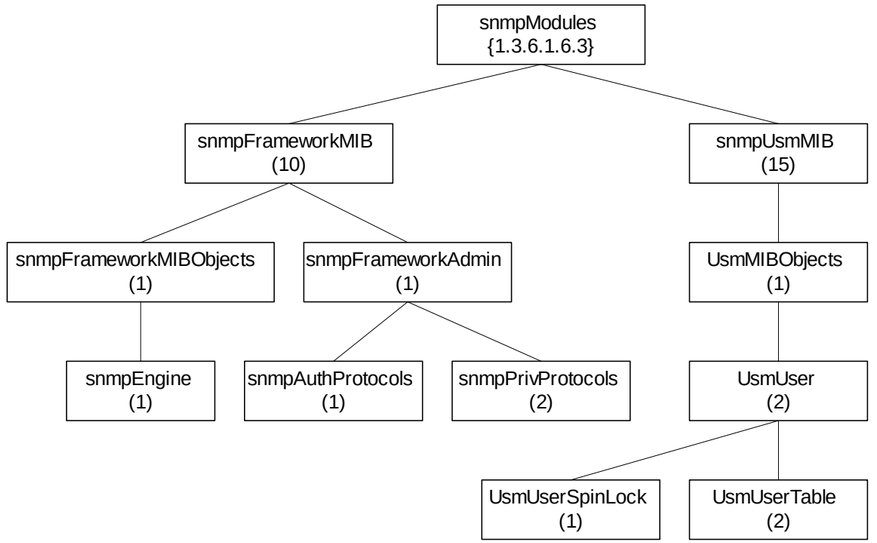


Figure 7.15 SNMPv3 MIB Objects for Security Parameters

Notes

Table 7.8 Security Parameters and Corresponding MIB Objects

Security Parameters	USM User Group Objects
msgAuthoritativeEngineID	snmpEngineID (under snmpEngine Group)
msgAuthoritativeEngineBoots	snmpEngineBoots (under snmpEngine Group)
msgAuthoritativeEngineTime	snmpEngineTime (under snmpEngine Group)
msgUserName	usmUserName (in usmUserTable)
msgAuthenticationParameters	usmUserAuthProtocol (in usmUserTable)
msgPrivacyParameters	usmUserPrivProtocol (in usmUserTable)

Privacy Module

- Encryption and decryption of scoped PDU (context engine ID, context name, and PDU)
- CBC - DES (Cipher Block Chaining – Data Encryption Standard) symmetric protocol
- Encryption key (and initialization vector) made up of secret key (user password), and timeliness value
- Privacy parameter is *salt* value (unique for each packet) in CBC-DES

Notes

Authentication Key

- Secret key for authentication
- Derived from user (NMS) ID and password
- MD5 or SHA-1 algorithm used
- Authentication key is *digest2*

Notes

Procedure:

1. Derive *digest0:*
   Password repeated until it forms 2^{20} octets.
2. Derive *digest1:*
   Hash *digest0* using MD5 or SHA-1.
3. Derive *digest2*:
   Concatenate authoritative SNMP engine ID and *digest1* and hash with the same algorithm

Authentication Parameters

- Authentication parameter is Hashed Message Access Code (HMAC)
- HMAC is 96-bit long (12 octets)
- Derived from authentication key (*authKey*)

Notes

Procedure:

1. Derive *extendedAuthKey*:
   Supplement authKey with 0s to get 64-byte string
2. Define *ipad, opad*, K1, and K2:
   ipad = 0x36 (00110110) repeated 64 times
   opad = 0x5c (01011100) repeated 64 times
   K1 = *extendedAuthKey* XOR *ipad*
   K2 = *extendedAuthKey* XOR *opad*
3. Derive HMAC by hashing algorithm used
   HMAC = H (K2, H (K1, *wholeMsg))*

Encryption Protocol

- Cipher Block Chaining mode of

  Data Encryption Standard (CBC-DES) protocol
- 16-octet *privKey* is secret key
- First 8-octet of *privKey* used as 56-bit DES key;

  (Only 7 high-order bits of each octet used)
- Last 8-octet of *privKey* used as pre-initialization vector

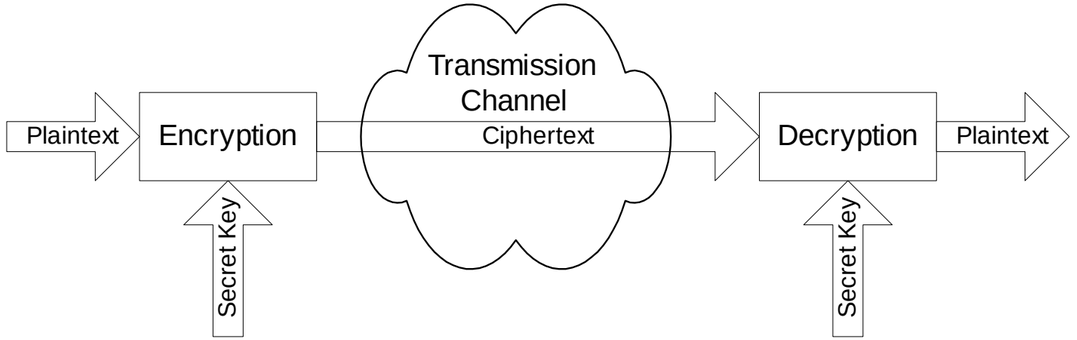


Figure 11.33 Basic Cryptographic Communication

Notes

- CBC Mode
  - Plaintext divided into 64-bit blocks
  - Each block is XOR-d with ciphertext of the previous block and then encrypted
  - Use pre-IV (initialization vector) for prefixing the first message block

Access Control

address access control, which deals with who can access network management components and what they can access. In SNMPv1 and SNMPv2, this subject has been covered using the community-based access policy. In SNMPv3, access control has been made more secure and more flexible. It is called VACM.

- **View-based Access Control Model**
  - Groups**: Name of the group** comprising security model and security name (In SNMPv1, is community name)
  - Security Level: 3 levels
    - no authentication - no privacy
    - authentication - no privacy
    - authentication - privacy
  - Contexts: Names of the context
  - MIB Views and View Families
    - MIB view is a combination of view subtrees
  - Access Policy
    - read-view
    - write-view
    - notify-view
    - not-accessible

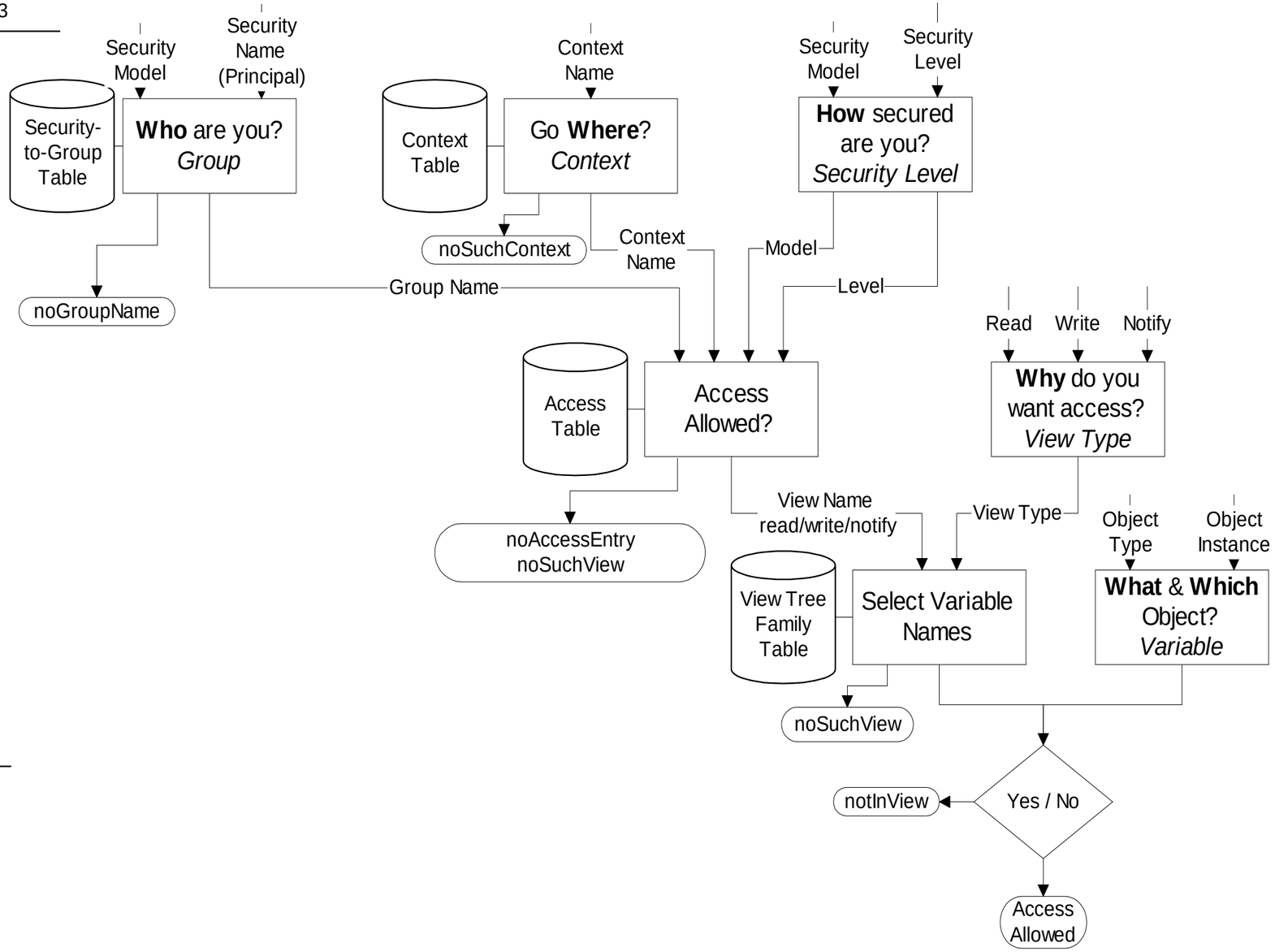
VACM gathers user and security model pairs into *security groups*, which provide a convenient means of identification.

VACM Process

Answers 6 questions:

1. Who are you (group)?
2. Where do you want to go (context)?
3. How secured are you to access the information (security model and security level)?
4. Why do you want to access the information (read, write, or send notification)?
5. What object (object type) do you want to access?
6. Which object (object instance) do you want to access?

VCAM Process



Notes

Figure 7.16 VACM Process

VACM MIB

Notes

- Four tables used to achieve access control:
  - Group defined by security-to-group table
  - Context defined by context table
  - Access determines access allowed and the view name
  - View tree family table determines the MIB view, which is very flexible

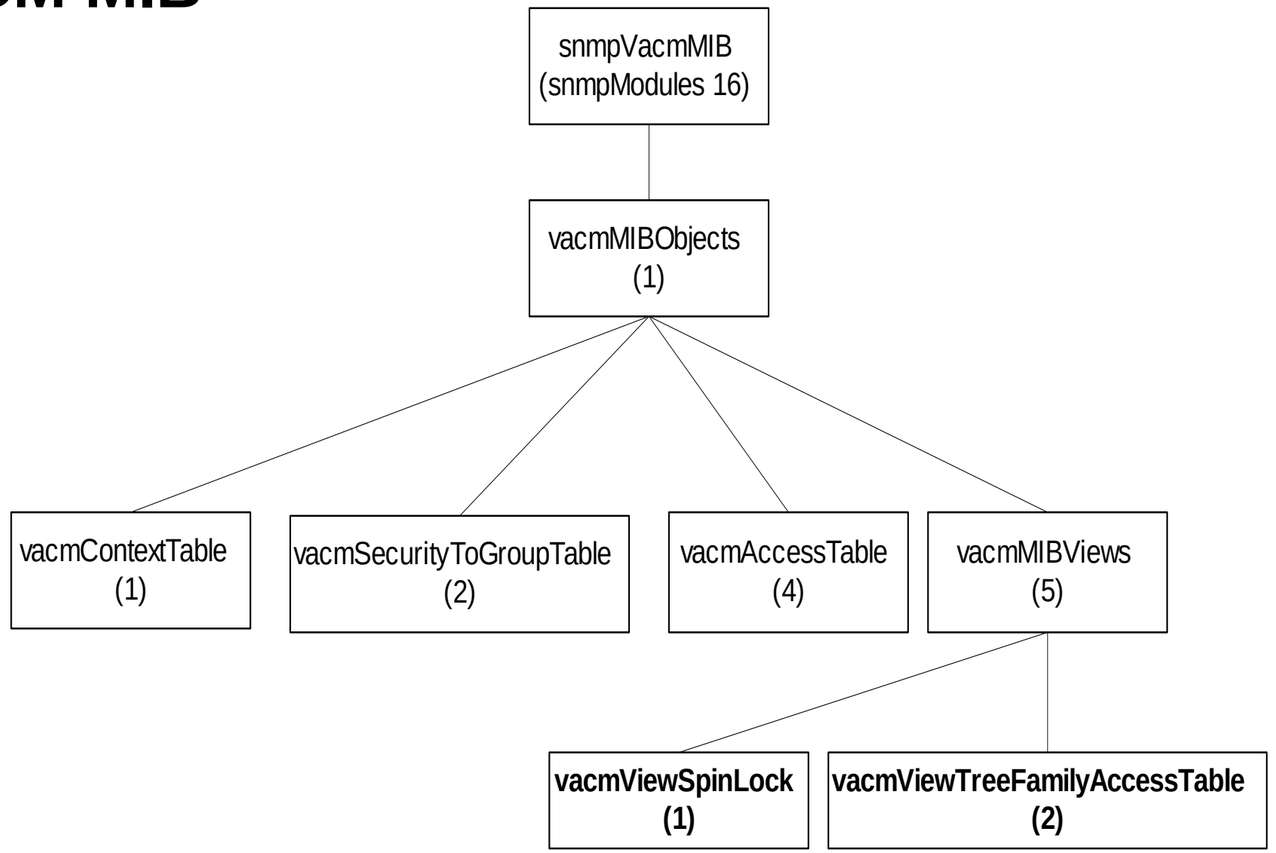


Figure 7.17 VACM MIB

MIB Views

- Simple view:
  - *system* 1.3.6.1.2.1.1
- Complex view:
  - All information relevant to a particular interface – *system* and *interfaces* groups
- Family view subtrees
  - View with all columnar objects in a row appear as separate subtree.
  - OBJECT IDENTIFIER (family name) paired with bit-string value (family mask) to select or suppress columnar objects

Notes

VACM MIB View

Notes

Example:

Family view name = “system”
Family subtree = 1.3.6.1.2.1.1
Family mask = “” (implies all 1s by convention)
Family type = 1 (implies value to be included)

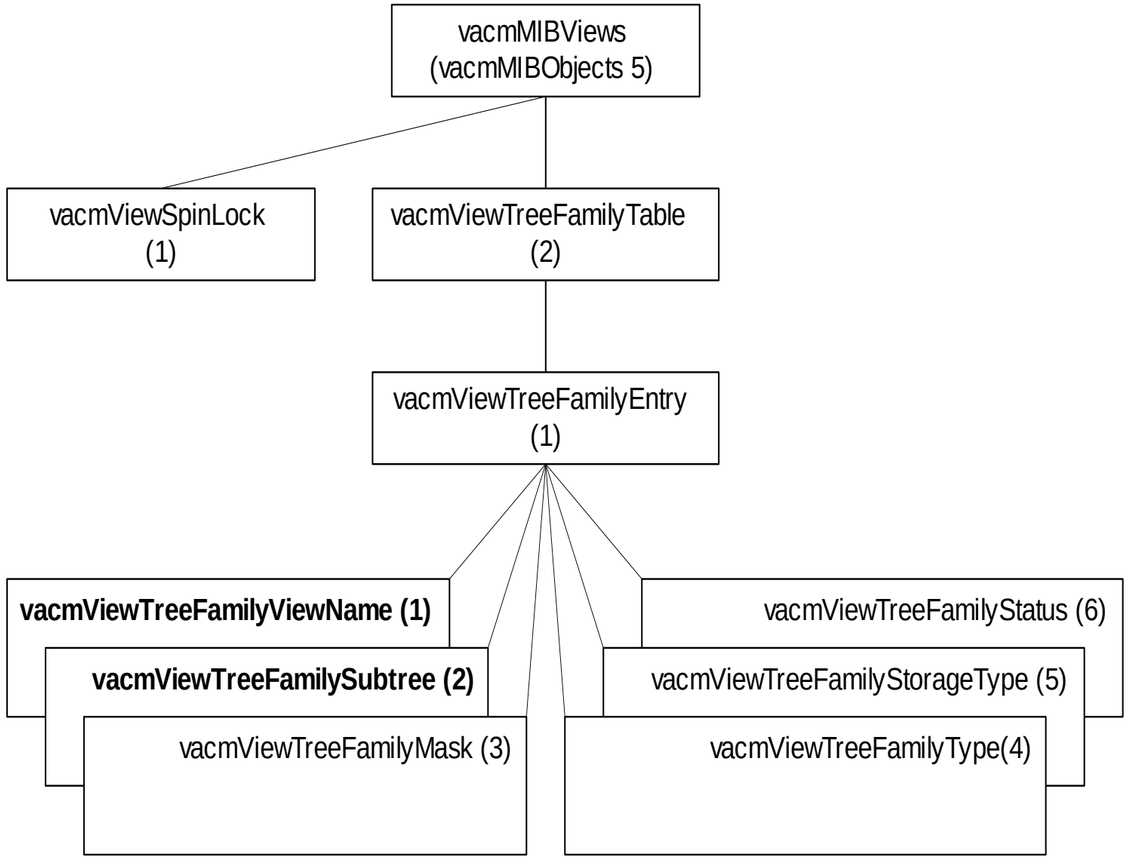


Figure 7.19 VACM MIB Views