Structure of Management Information

- ‘Structure of Management Information' (SMI) defines the general framework within which MIBs can be defined and constructed (STD58)
- SMI identifies data types usable in a MIB and specifies how resources within a MIB are represented & named
  
  SNMP ‘philosophy’ is to be simple & easy to implement
  
  => SMI only supports simple data types
  
  i.e. scalars and two-dimensional arrays (tables)

- SMI provides:
  - standardised technique for definition structure of a particular MIB
  - standardised technique for definition individual objects, (including object syntax & values)
  - standardised technique for encoding object values

1 In OSI/3GPP it is possible to have more complex data/object structures
MIB Structure

- Managed Objects arranged in hierarchical structure to allow object type identification
- Leaf objects of the tree are the actual managed objects, each of which represent some resource, activity or related information to be managed
- The tree structure itself defines a grouping of objects into logically related sets
- Each object type has an Object Identifier to uniquely identify object
- Because of the value associated with the type (i.e. object identifier), the naming convention also serves to identify the structure of object types
e.g. 1.3.6.1.2.1.4 indicates the root of the IP subsystem for that device

MIB Hierarchy for Object Identification
MIB Object Syntax

- Each object defined in formal way (using a subset of ASN.1)
- Definition of object syntax includes the specification of the data type of the object, its allowable forms and value ranges, & its relationship to other objects within the MIB
- Data types supported in MIBs are:
  - Integer
  - Octet string
  - NULL
  - Object identifier (OID)
  - Sequence, sequence of (used for tables)
  - Application-wide data types
Application-wide data types

- **network address**: currently only one specified, namely ipaddress (a 32 bit address)
- **counter**: (or rollover counter) is a non negative integer that may be incremented but not decremented. Max values of $2^{32-1}$. When counter reaches its maximum, it wraps around & starts increasing again from zero
- **gauge**: a non negative integer that can increase & decrease, with maximum value of $2^{32-1}$. If max reached, gauge remains latched at that value until reset
- **timeticks**: a non negative integer counts the time in hundredths of a second since some epoch. When object defined in MIB, epoch is identified e.g. start-up or re-initialization. Is NOT supported by time sync. Protocol.
- **opaque**: supports the ability to pass arbitrary data. Data is encoded as octet string for transmission. Data format itself may be defined by another syntax type.
Defining Objects in a MIB

- Each managed object has an object type (defined in ASN.1)
- Key components of an Object Type:
  - **Syntax**: must resolve to an instance of one of the object Syntaxes mentioned earlier
  - **Access**: defines the way in which an instance of an object may be accessed (via SNMP). Specifies the minimum level of support required for that object type. Options are *read-only*, *read-write*, *write-only* & *not-accessible*
  - **Status**: Indicates implementation support for object (mandatory or optional or deprecated). Deprecated is one which must be supported but which will most likely be removed from the next version of the MIB
  - **Description**: a textual description of the semantics of the object type (optional)
Example Object Type definition

Creating the SMI tcpMaxConn Object Type

tcpMaxConn  OBJECT-TYPE
   SYNTAX  INTEGER
   ACCESS read-only
   STATUS mandatory
   DESCRIPTION
   "The limit on the total number of TCP connections the entity can support. In entities where the maximum number of connections is dynamic, this object should contain the value -1"

   ::= {tcp 4}

Hence the final OID of this object-type in the MIB is 1.3.6.1.2.1.6.4
Other elements of Object Type definition

- **ReferPart**: a textual cross reference to an object defined in some other MIB module (optional)
- **IndexPart**: used in defining tables. This clause may be present only if the object type corresponds to a conceptual row.
- **DefValPart**: defines a default value that may be used when an object instance is created (at the discretion of the agent). (optional)
- **Value Notation ( ::= )**: indicates the name used to access the object via SNMP
Defining Tables

• SMI supports only one form of structured data: a simple two-dimensional table
• Definition of tables involves using sequence and sequence of ASN.1 types and IndexPart of the Object-Type definition

Example:
• Consider the TcpConnectionEntry managed object which contains information about a particular current TCP connection and contains tcp connection state, tcp connection local address, tcp connection local port, tcp connection remote address, tcp connection remote port
• A management station maintains a table of such TCP connection entries. Thus each entry in this table (tcpConnTable) represents the state information stored in the managed station for one connection.
Example cont.

• The overall table consists of a sequence of (zero or more) TcpConnEntry
• Each row consists of a sequence that includes five scalar elements (Integer, Ipaddress, Integer, Ipaddress, Integer)
• The Index component determines which object value(s) will be used to distinguish each row in the table
• Note: Nesting of tables (i.e. an entry in a table is itself a table) is NOT allowed in SNMP
Table Example

netSnmpHostsTable OBJECT-TYPE
SYNTAX SEQUENCE OF NetSnmpHostsEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION "An example table that implements a wrapper around the /etc/hosts file on a machine using the iterator helper API." ::= {
netSnmpExampleTables 2 }

netSnmpHostsEntry OBJECT-TYPE
SYNTAX NetSnmpHostsEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION "A host name mapped to an ip address"
INDEX { netSnmpHostName } ::= { netSnmpHostsTable 1 }

NetSnmpHostsEntry ::= SEQUENCE { netSnmpHostName OCTET STRING, netSnmpHostAddressType InetAddressType, netSnmpHostAddress InetAddress, netSnmpHostStorage StorageType, netSnmpHostRowStatus RowStatus }
Identifying Object Instances

• Can identify an Object Type by using the object identifier (defined by position of the object in the MIB tree),
• How do we identify object instances?
• Two kinds of object instances:
  • scalar objects
  • columnar objects (i.e. objects that occur in a table)
• For columnar objects, two techniques for identifying instances
  – serial-access - based on lexicographic ordering of instances in MIB
  – random access - identification of a specific columnar object in a table
Identifying Scalar Objects

• There is no ambiguity between an object type and an instance of that type since there is only one object instance for each scalar type

• however, for consistency, (& to distinguish between object type and object instance) an instance identifier of a nontabular scalar object consists of its object identifier concatenated with 0
Identifying Columnar objects: Random Access technique

• When a table is defined, one or more objects which make up the elements of a row are specified as INDEX objects
• these INDEX objects are used to distinguish one row from another
• Combination of object identifier for a columnar object and one set of values of the INDEX objects specifies a particular scalar object in a particular row of a table
• SNMP concatenates the scalar object identifiers with the values of the INDEX objects, listed in the order in which the INDEX objects appear in the table definition
Converting Object instance values into subidentifiers

For each INDEX object instance:

• integer-valued: A single subidentifier takes the integer value (valid only for nonnegative integers)

• string-valued, fixed length: each octet of the string is encoded as a separate subidentifier, for a total of n subidentifiers for a string of length n octets

• string-valued, variable length: for a string of length n octets, the first subidentifier is n; this is followed by each octet of the string encoded as a separate subidentifier, for a total of n+1 subidentifiers

• object-identifier-valued: for an object identifier with n subidentifiers, the first subidentifier is n, this is followed by the value of each subidentifier in order, for a total of n+1 subidentifiers

• Ipaddress-valued: these are four subidentifiers, in the familiar a.b.c.d notation
Ambiguities in row identification

- Theoretically ambiguities in identifying a row should not occur as INDEX object values should unambiguously distinguish a conceptual row.
- However in MIB I this is not always the case.
- However in MIB II any new table objects conform to the definition of INDEX and any existing tables which suffer from this ambiguity are deprecated.
Identifying Columnar objects: Lexicographic Ordering

- Ordering of the components of an object identifier can be generated by traversing the tree of object identifiers in the MIB.

- This ordering extends to object instance identifiers since an object instance identifier is also a sequence of integers.

- This ordering can be used by a management station which does not know the exact makeup of the MIB view supported by an agent.

- In this case the management station needs a way of searching for and accessing objects without specifying their name.

- By using lexicographic ordering, the management station can supply an object or object instance identifier and then ask for the object instance that occurs next in the ordering.

i.e. given a particular position in the MIB tree the management station can then traverse the MIB with a sequence of get-next retrieval operations.
Caveats and Limitations of SNMP MIBs

- Evidence has shown that although vendors claim to support SNMP agents (and managed objects), their accuracy of measurements can be inconsistent.
- There are limitations in using SNMP MIBs e.g. sophistication & representative power for expressing managed objects.
- MIB-II (RFC 1212) provides a ‘improved’ and more concise set of MIB definition from the original MIB definitions (RFC 1155).
- Many standard MIBs lack writeable objects (for CM).
SNMP v1 Agents and Security

- Many Management stations may access/manage an agent => each agent must control the use of that MIB by one (or more) management stations
  - Agent control consists of the following security issues:
    - Authentication service: agent can limit access to MIB to authorised management stations
    - Access Policy: agent controls different access privileges to different management stations
    - Proxy service: agent may act as proxy to other managed stations (i.e. may implement authentication service and access policy for other managed systems on proxy system)
Authentication Service

- Ensures the recipient that the received message is from the source from which it claims to be
- SNMP v1 authentication is trivial!
- Every (Get or Set) request from a management station includes a community name
- Message is assumed authentic if the sender knows the appropriate community name (for management station/agent pairing)

=> Community Name used as a password between management station and agent!
SNMP V2

• SNMP V1 only provides limited security based on concept of community
• SNMP V2 went through several revisions to provide new security and access control models
  – SNMP V2: Changes to PDU formats, and other extensive protocol changes. This was largely rejected by the community
  – SNMP V2c: Community model – lighter-weight revision of SNMPv1
  – SNMP V2u: User model – user-oriented model, adapted and adopted
• SNMP v3: Authentication, Privacy, Access Control
Access Policy

• Agent typically limits access to its MIB to a selected set of management stations

• Agent can provide different categories of MIB access to different management stations by:
  – limiting the management station(s) view of the MIB to a specific subset of objects
  – limiting the access mode for the community (either read-only or read-write) and is applied to all objects in a communities MIB view

• Combination of MIB view and access mode is referred to as SNMP ‘profile’
# Relationship between MIB access category and SNMP access Mode

<table>
<thead>
<tr>
<th>MIB Access Category</th>
<th>SNMP Access Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read-only</td>
<td>READ-ONLY Available for get and trap operations</td>
</tr>
<tr>
<td>Read-write</td>
<td>Available for get and trap operations</td>
</tr>
<tr>
<td>Write-only</td>
<td>Available for get &amp; trap operations, but the value is implementation specific</td>
</tr>
<tr>
<td>Not accessible</td>
<td>Unavailable</td>
</tr>
</tbody>
</table>
Limitations of SNMPv1

• Not suitable for management of very large networks (principally because of it’s reliance on polling)
• Not suited for retrieval of large volumes of data
• SNMP traps are not acknowledged
• authentication is trivial
• SNMP does not directly support imperative commands (actions)
• SNMP MIB is limited and does not readily support applications
• SNMP does not support manager to manager communications
• Many of above deficiencies are addressed in SNMPv3
SNMP V3 Security

• Introduce three important services
  • Authentication
  • Privacy
  • Access Control

*Data Encryption Standard
SNMP V3 Architecture

- SNMPv3 is defined in a modular fashion
- Each SNMP entity includes a single SNMP engine
- Engine implements functions for
  - Sending and receiving messages
  - Authenticating and encrypting/decrypting messages
  - Controlling access to managed objects
- Role of entity determines what modules are implemented
- Can define different versions for each module (e.g. SNMPv4!)
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.

**Authoritative Engine**
- This is decided based on the following rules:
  
  - When SNMP message contains a payload that expects a response => the receiver 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.
    • Authoritative engine sends a message, it contains the current value of its clock
      • Non-authoritative recipient synchronizes on that clock.
      • E.g. authoritative sends Trap, Response, Report

  • Each non-authoritative engine maintains an estimate of the time value for each authoritative 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-authoritative 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.
SNMP V3 User-Based Security Model

• The security-related parameters include the following:
  
  • msgAuthoritativeEngineID: The snmpEngineID of the authoritative SNMP engine involved in
    the exchange of this message.

  • msgAuthoritativeEngineBoots: The snmpEngineBoots value of the authoritative SNMP
    engine involved in the exchange of this message. The object snmpEngineBoots is an integer in
    the range 0 through 2^31 — 1 that represents the number of times that this SNMP engine has
    initialized or reinitialized itself since its initial configuration.

  • msgAuthoritativeEngineTime: The snmpEngineTime value of the authoritative SNMP engine
    involved in the exchange of this message. The object snmpEngineTime is an integer in the 0
    through 2^31 — 1 range that represents the number of seconds since this authoritative SNMP
    engine last incremented the snmpEngineBoots object.
    
    • Each authoritative SNMP engine is responsible for incrementing its own snmpEngineTime value once
      per second.
    • A non-authoritative engine is responsible for incrementing its notion of snmpEngineTime for each remote
      authoritative engine with which it communicates.

  • msgUserName: The user (principal) on whose behalf the message is being exchanged.

  • msgAuthenticationParameters: Null if authentication is not being used for this exchange;
    otherwise, this is a privacy parameter. In USM, the privacy parameter is a parameter used in
    the encryption algorithm DES.
Privacy Using Conventional Encryption

- The authentication mechanism in SNMPv3 assures received message was transmitted by the principal whose identifier appears as the source in the message header.
  - And that the message was not altered in transit and that it was not artificially delayed or replayed.

- To achieve authentication:
  - Principal and remote SNMP share a secret authentication key.
  - Sending entity provides the authentication code with the SNMPv3 message it is sending.
  - This code is a function of the contents of the message, the identity of the principal and engine, the time of transmission, and a secret key that should be known only to the sender and the receiver.
  - The secret key must initially be set up outside of SNMPv3 as a configuration function.
    - Network manager is responsible for distributing initial secret keys to be loaded into the databases of the various SNMP managers and agents.
    - This can be done manually or by using some form of secure data transfer outside of SNMPv3.
View-Based Access Control Model

• The access control facility makes it possible to configure agents to provide different levels of access to the agent's MIB to different managers.

• An agent entity can restrict access to its MIB for a particular manager entity in two ways.
  • First, it can restrict access to a certain portion of its MIB.
  • Second, the agent can limit the operations that a principal can use on that portion of the MIB.

• The access control policy to be used by an agent for each manager must be preconfigured; it essentially consists of a table that details the access privileges of the various authorized managers.

• Unlike authentication, which is done by user, access control is done by group, where a group may be a set of multiple users.
<table>
<thead>
<tr>
<th>Version</th>
<th>Level</th>
<th>Authentication</th>
<th>Encryption</th>
<th>What Happens</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>NoAuth NoPriv</td>
<td>Community String</td>
<td>None</td>
<td>Uses a community string match for authentication</td>
</tr>
<tr>
<td>V2c</td>
<td>NoAuth NoPriv</td>
<td>Community String</td>
<td>None</td>
<td>Uses a community string match for authentication</td>
</tr>
<tr>
<td>V3</td>
<td>NoAuth NoPriv</td>
<td>username</td>
<td>None</td>
<td>Uses a username match for authentication</td>
</tr>
<tr>
<td>V3</td>
<td>Auth NoPriv</td>
<td>MD5 or SHA</td>
<td>None</td>
<td>Provides authentication based on the Hash-Based Message Authentication Code (HMAC) Message Digest 5 (MD5) algorithm or the HMAC Secure Hash Algorithm (SHA).</td>
</tr>
<tr>
<td>V3</td>
<td>Auth Priv</td>
<td>MD5 or SHA</td>
<td>DES</td>
<td>Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms. Provides Data Encryption Standard (DES) 56-bit encryption in addition to authentication based on the Cipher Block Chaining (CBC) DES (DES-56) standard</td>
</tr>
</tbody>
</table>
SNMP Question

• You are analysing a single LAN which has one management station which is responsible for managing a network of agents. Each agent is to be polled every 30 minutes. It takes 99ms to process a query, with a network delay of about 2ms.
  a) What is the maximum number of agents that the management station can handle when engaged in full-time polling?
  b) If a single poll takes 1KB of bandwidth, and the overall capacity of the network is 100MB per hour, what is the effect of your polling scheme on utilisation at full-time polling?
  c) How might you reduce this cost?
SNMP V3 Principals

• Introduces concept of principal which is the entity on whose behalf services are provided or processing takes place

• Principal operates from management station and issues SNMP commands to agent systems

• Identity of principal and target together determine Authentication, Privacy and Access Control

• The use of principals allows security policies to be tailored to the specific principal, agent, and information exchange