### Part 2: Structure of Management Information Version 2 (SMIv2)

The SMI is the “Data Definition Language”, based loosely on an old version of ASN.1 and used to model and define management information to be exchanged between an SNMP agent and manager.

- ASN.1 Background
- Data Types available in SMIv2
- SMIv2 Macros

---

### ASN.1 Background: Definition

- **Abstract Syntax Notation One (ASN.1)** is a specification of a *rich* grammar or notation and encoding rules that:
  - enable complicated types to be defined and enables values of these types to be specified; and
  - define encoding rules for values of ASN.1 types that completely specify the representation of values of that type during transfer, i.e., a transfer syntax.

- **Understanding SNMP MIBs**, by Perkins and McGinnis. Prentice-Hall. ISBN 0134377087

---
SNMP Protocol Encoding

- The goal is to transfer (management) information between heterogeneous systems via a well-understood and machine-independent syntax
- Machines vary by
  - word size: e.g., 8, 16, 32, 64, other (weirdo)
  - arithmetic type: 1’s complement, 2’s complement, BCD
  - byte ordering: big endian, little endian
  - character set: ASCII, EBCDIC, 16 bit characters

ASN.1 Encoding: Goal

- This is a classic presentation layer problem and is addressed by ISO standards: ASN.1 (IS 8824-8825)

A carefully engineered subset of the ISO standard presentation protocol (ASN.1) is used to meet this goal.
Universal Types: NULL, INTEGER, OCTET STRING

- **NULL** is commonly used as a place-holder for values.
- **INTEGER** is a simple type with distinguished values that are the positive and negative whole numbers, including zero.
  - The largest integer supported by the SMI has a maximum length of four bytes (32 bits).
- **OCTET STRING** is a simple type which is an ordered sequence of zero or more octets (< 64k)

Managed Object with INTEGER SYNTAX Example

```plaintext
numChildren OBJECT-TYPE
SYNTAX INTEGER (0..65535)
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of children in the Child database on this system."
 ::= { childCharacteristics 1 }
```
Managed Object with OCTET STRING
SYNTAX Example (RFC 1759)

prtGeneralCurrentOperator OBJECT-TYPE
SYNTAX OCTET STRING (SIZE(0..127))
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"The name of the current human operator responsible for operating this printer. ..."
::= { prtGeneralEntry 4 }

Universal Type: OBJECT IDENTIFIER (OID)

◆ Primary Purposes of OIDs:
  ○ To uniquely and globally name management information
  ○ To provide a way of delegating administrative authority.
◆ Four written OID forms for human convenience:
  ○ NameForm
  ○ NumberForm
  ○ NameAndNumberForm
  ○ PrefixedNumberForm
**OBJECT IDENTIFIER Semantics**

- The semantics of OBJECT IDENTIFIER values are defined by reference to an OBJECT IDENTIFIER tree.
- Each arc of the tree is labeled by an OBJECT IDENTIFIER component, which is a numeric value.

---

**Basic OBJECT IDENTIFIER Tree**

[Diagram of the Basic OBJECT IDENTIFIER Tree]

---

David Partain (david.partain@ericsson.com)
OBJECT IDENTIFIER Examples

internet OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }
mgmt OBJECT IDENTIFIER ::= { internet 2 }
mib OBJECT IDENTIFIER ::= { mgmt 1 }
system OBJECT IDENTIFIER ::= { mib 1 }
sysObjectID OBJECT IDENTIFIER ::= { system 2 }

which, when concatenated, yields
sysObjectID OBJECT IDENTIFIER ::= { iso org(3) dod(6) internet(1) mgmt(2) mib(1) system(1) 2 }

which can be written numerically as: 1.3.6.1.2.1.1.2

Universal Types: SEQUENCE Type

◆ A SEQUENCE is an ordered list of values, used for defining rows of a table as SEQUENCEs of columns.
◆ Used to generate a list in an SNMP MIB.
◆ The syntax takes the form:
  
  SEQUENCE { <type1>, ..., <typeN> }

  where each <type> resolves to one of the ASN.1 primitive types listed previously (INTEGER, OCTET STRING, etc.).
**SEQUENCE Example**

ChildEntry ::=  
  SEQUENCE {  
    childIndex INTEGER,  
    childName HumanName,  
    childBirthDate DateOfBirth,  
    childHairColor INTEGER,  
    childLikesToCleanRoom TruthValue  
  }

**Universal Types: SEQUENCE-OF Type**

- The SEQUENCE-OF is a structured type, defined by referencing a single existing type.
- The SMI uses the SEQUENCE-OF type for defining tables as a SEQUENCE-OF SEQUENCEs.
- For table construction, it takes the form
  
  SEQUENCE OF < entry>  
  where < entry> resolves to a list constructor (SEQUENCE).
- Lists and tables are sometimes referred to as aggregate types.
Brief Introduction to the Internet Standard Management Framework

SEQUENCE OF Example

childTable OBJECT-TYPE
SYNTAX SEQUENCE OF ChildEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION "...."
::= { childCharacteristics 2 }

Summary: ASN.1 Simple and Constructor Types used in SNMP

INTEGER (-2147483648..2147483647)
Integer32
OCTET STRING (SIZE (0..65535))
OBJECT IDENTIFIER
NULL
SEQUENCE
SEQUENCE OF
### SNMP-Specific Defined Types

- IPAddress
- Counter32
- Gauge32
- TimeTicks
- Opaque
- Counter64
- Unsigned32
- BITS

---

#### ifInUcastPkts OBJECT-TYPE

**SYNTAX** Counter32  
**ACCESS** read-only  
**STATUS** current  
**DESCRIPTION**

"The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were not addressed to a multicast or broadcast address at this sub-layer."

::= { ifEntry 11 }

---

David Partain (david.partain@ericsson.com)
Brief Introduction to the Internet Standard Management Framework

TimeTicks Example

```plaintext
sysUpTime OBJECT-TYPE
SYNTAX      TimeTicks
ACCESS      read-only -- MAX-ACCESS in SMIv2
STATUS      mandatory -- current in SMIv2
DESCRIPTION
"The time (in hundredths of a second) since the
network management portion of the system was
last re-initialized."
::= { system 3 }
```

SNMP Information Modules

- Usually three kinds:
  1. MIB modules (objects and notifications)
  2. compliance statements
  3. agent capability statements
- A “standard” information module might contain object definitions and a compliance statement
- An “enterprise-specific” information module might contain object definitions; and, several capability statements
Invoking Macros

- Basic syntax is:
  
  \[ \text{<descriptor> <macro> <clauses>} \rightarrow \text{<value>} \]

  e.g.,
  
  ```
  sysUpTime OBJECT-TYPE
  SYNTAX      TimeTicks
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
  "The time (in hundredths of a second) since the network
  management portion of the system was last re-initialized."
  ::= { system 3 }
  ```

- The SMIv2 limits the length of descriptors to 64 (and doesn't allow hyphens or underscores)

---

MODULE-IDENTITY Macro

```
snmpVacmMIB MODULE-IDENTITY
LAST-UPDATED "9901200000Z" -- 20 Jan 1999, midnight
ORGANIZATION "SNMPv3 Working Group"
CONTACT-INFO "WG-email: snmpv3@tis.com
Subscribe: majordomo@tis.com
In message body: subscribe snmpv3"

DESCRIPTION "The management information definitions for
the View-based Access Control Model for SNMP."
REVISION "9901200000Z" -- 20 Jan 1999, midnight
DESCRIPTION "Clarifications, published as RFC2575"
REVISION "9711200000Z" -- 20 Nov 1997, midnight
DESCRIPTION "Initial version, published as RFC2275"
 ::= { snmpModules 16 }
```
SMIv2 TEXTUAL-CONVENTION Macro

- TEXTUAL-CONVENTION MACRO associates
  - DISPLAY-HINT (e.g., DISPLAY-HINT "1x:"
  - STATUS clause
  - DESCRIPTION clause
  - optional REFERENCE clause
with an OBJECT SYNTAX

Some Defined Textual Conventions

- DisplayString: NVT ASCII string up to 255 characters
- TruthValue: true(1) or false(2)
- TestAndIncr: provides for atomic, or sequenced, operations
- RowStatus: for conceptual row creation
- DateAndTime: A date-time specification (timezone information is optional)
Using TEXTUAL-CONVENTION Macro

DisplayString ::= TEXTUAL-CONVENTION
   DISPLAY-HINT "255a"
   STATUS  current
   DESCRIPTION "Represents textual information taken from the
                 NVT ASCII character set, as defined in pages 4,
                 10-11 of RFC 854. Any object defined using this
                 syntax may not exceed 255 characters in length."
   SYNTAX   OCTET STRING (SIZE ( 0..255 ) )

Traps and Informs:
NOTIFICATION-TYPE Macro

◆ Traps report “exceptional events”
◆ Traps are unacknowledged, informs are acknowledged. Both called “notifications”.
◆ They are used sparingly in the Internet-standard Management Framework
◆ Notifications defined using the NOTIFICATION-TYPE macro
NOTIFICATION-TYPE Example

linkUp NOTIFICATION-TYPE
OBJECTS { ifIndex }
STATUS current
DESCRIPTION
"A linkUp trap signifies that the SNMPv2 entity, acting in an agent role, recognizes that one of the communication links represented in its configuration has come up."
::= { snmpTraps 4 }

OBJECT-TYPE Macro Clauses

- **Purpose** is to define semantics and hang the object in the MIB tree!
- **SYNTAX** clause is restricted to “simple” things:
  - Simple and application-specific types
  - refinements: enumerated integers, size ranges, etc.
- **MAX-ACCESS** clause is “what makes protocol sense” independent of authorization policy:
  - not-accessible, accessible-for-notify, read-only, read-write, read-create
- **STATUS** clause
  - (almost) always current
OBJECT-TYPE Macro Clauses (Continued)

- DESCRIPTION clause mandatory
- REFERENCE, UNITS, DEFVAL clauses
- For conceptual rows:
  - INDEX clause tells how instances are identified; or,
  - AUGMENTS clause identifies a row “extension”
    (exactly one clause must be used)

---

OBJECT-TYPE Macro Example

```plaintext
childName OBJECT-TYPE
SYNTAX OCTET STRING (SIZE (0..255))
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"Indicates the name of the child."
 ::= { childEntry 2 }
```

---
Naming Management Information: Scalars

- Naming a piece of management information requires a class name
  - e.g., sysContact (or 1.3.6.1.2.1.1.1)
- as well as an instance sub-identifier of that class.
- Scalar objects are those for which there is one and only one instance of an object, and for these the instance sub-identifier is always 0
  - e.g., to name the one and only one instance of sysContact, you would use sysContact.0 (or 1.3.6.1.2.1.1.1.0)

Naming Management Information: the INDEX Clause for MIB Tables

- Defines the algorithm for instance identification by naming one or more indexing objects (keys) into a table
- Indices can be INTEGERs, OCTET STRINGs, or OBJECT IDENTIFIERS
- Naming management information in a table is then done using the object class OBJECT IDENTIFIER concatenated with OID subidentifiers for the key(s).
- Full details of how instances are built is beyond the time we have together, but examples are useful...
Some INDEX Examples

ifEntry OBJECT-TYPE
... 
INDEX { ifIndex } <======= NOTE! the key

ifIndex OBJECT-TYPE
SYNTAX InterfaceIndex
... 
 ::= { ifEntry 1 }

ifDescr OBJECT-TYPE
SYNTAX DisplayString (SIZE (0..255))
... 
 ::= { ifEntry 2 }

What OID identifies the description of the 3rd interface? ifDescr.3

Some INDEX Examples (cont.)

serverUDPAppEntry OBJECT-TYPE
... 
INDEX { serverIP, serverUDPPort }

serverIP OBJECT-TYPE serverUDPPort
SYNTAX IpAddress SYNTAX Unsigned32
... 
 ::= { serverUDPAppEntry 1 } ::= { serverUDPAppEntry 2 }

serverUDPAppName OBJECT-TYPE
SYNTAX DisplayString (SIZE (0..32))
... 
 ::= { serverUDPAppEntry 3 }

What OID identifies the description of the app running at 12.0.0.1 port 3488? serverUDPAppName.12.0.0.1.3488