An Application of XML on SNMP

Pipat Hiranvanichakorn[1) , Prakan Puvibunsuk[2)

[1) Associate professor, School of Applied Statistics, National Institute of Development Administration.

[2) Graduate student, School of Applied Statistics, National Institute of Development Administration.

Abstract

This paper reports a study of an application of the eXtensible Markup Language (XML) on the Simple Network Management Protocol (SNMP) model. In the conventional model, Abstract Syntax Notation 1 (ASN.1) has been used in SNMP to describe management information and encode data transferred between devices. However, ASN.1 is large, complex and not especially efficient. In this paper, XML which is widely used as data exchange standard in the Internet, is proposed instead of ASN.1. In the paper, data in the Management Information Base (MIB) are described by using XML. The SNMP protocol is used to interact between the management station and managed nodes. The data sent between the management station and managed nodes are in the form of XML. With XML, the data is easily manipulated and is described precisely, so the web application in the management station can work with the data intelligently. Furthermore, this management information can be transferred to other applic ations efficiently.

1. Introduction

Simple Network Management Protocol (SNMP) has been widely used as a well known standard for monitoring and managing a computer network. The SNMP model consists of four components, the managed nodes, the management stations, management information and a management protocol, as depicted in Fig.1. Each managed node has a agent which maintains management information. Management information describes the agent's state and history and affect the agent's operation. Management stations use the management protocol to communicate with managed nodes. A management station issues commands to the managed nodes to read and write management information, and to control some operations of the managed nodes.

The heart of the SNMP model is the management information. All management variables are defined in a data structure called Management Information Base (MIB). In order to be able to transfer information between multivender devices, Abstract Syntax Notation 1 (ASN.1) defined by OSI is used in SNMP to describe management information and encode the transferred data. However, ASN.1 is large, complex and not especially efficient.[1] It uses some encoding rules to minimize the number of bits on the wire, at the cost of wasting CPU time at both ends. Therefore, ASN.1 is rarely used for representing and transferring information through a network.

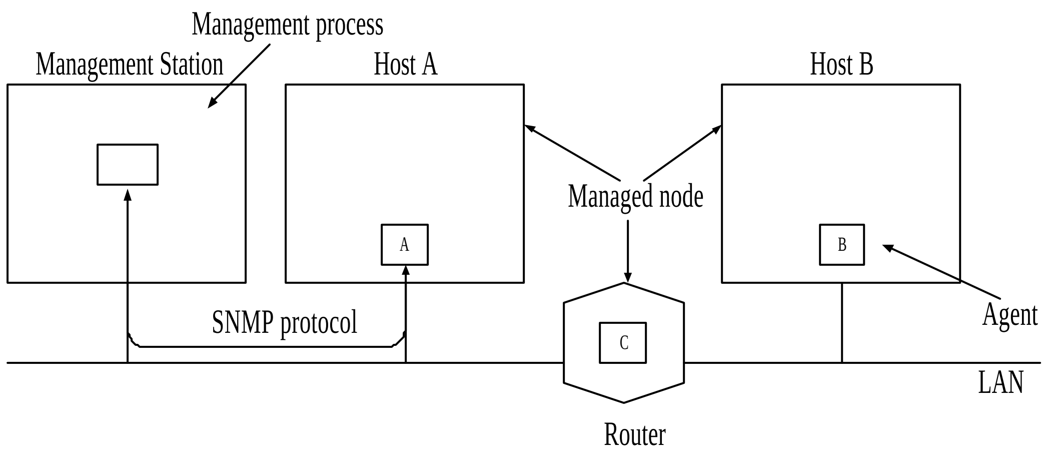


Fig.1 SNMP model

Nowsadays, the eXtensible Markup. Language (XML) is widely used as data exchange standard in the Internet. XML is text–based, so it makes XML documents both human–readable and computer manipulatable. Because an XML document describes data precisely, it can be processed by any application. It is considered be able to describe data of virtually any type in a structured manner.[2]

In this paper, data in the Management Information Base (MIB) are described by using XML in stead of ASN.1. The SNMP protocol is used to interact between the management station and managed nodes. Requests sent by the management station are in the form of XML. After getting the requests the agents do the work, and the reponses are sent back in the form of XML. With XML, the data is described precisely and easily manipulated, so the web application in the management station can work with the data intelligently. Furthermore, this management information can be transferred to other applications efficiently.

2. Network Management System

Fig. 2 depicts the network management system. A network manager gets into the system through a web browser on a PC. The manager then connects to the network management web server and identifies himself (herself) by using SSL protocol. After the authentication process, the network management software in the form of Active X is downloaded into the management station. The management process connects to the agent in the managed node by using TCP/IP. When the manager issues a command, the request is sent in the form of XML to the agent. When the request arrives, the agent verifies the request with the MIB. If the request is valid, the agent does the work and the response is sent back in the form of XML to the manager process. This request–reponse is sent according to the SNMP protocol. The XML response is then parsed, and the information is extracted. This information is can be presented to the management or sent for further processing.

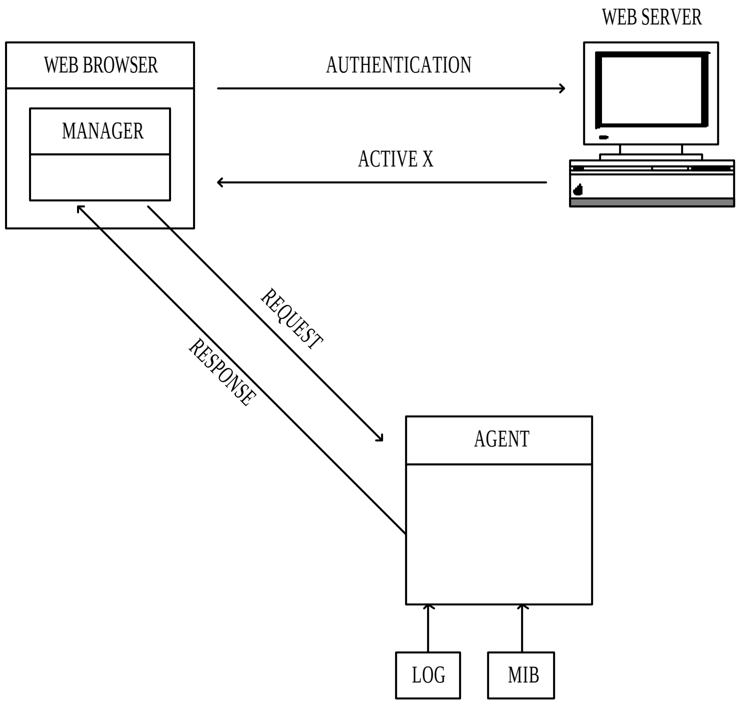


Fig.2 A network management system

3. Management Information

The heart of the SNMP model is the management information. It is used to describe the state, history and affect in the operation of a managed device. In order to be able to communicate between multivendor, this management information must be rigidly specified. Furthermore, a standard way is needed to encode the information for transfer between different devices. In SNMP model, all management objects (variables) in a network are defined in a data structure called Management Information Base (MIB). In the following sections, we describe how the management objects are defined by using ASN.1. Then, we describe how ASN1 is replaced by XML. Furthermore, we describe how the transferred data are encoded by using ASN1 transfer syntax. Then, we describe how the XML data are transferred.

3.1 Management Information Base

Fig.3 shows an example of describing the structure of the management information in the MIB by using ASN.1. Each SNMP variable has four required parameters. The SYNTAX parameter defines the variable's data type. The MAX-ACCESS one tells the information about the variable's access. It may have values of read-write, read only, not accessibly or write-only. The STATUS tells that the variable is conformant with the current SNMP specification or not. It has three values, i.e., current, obsolete and deprecated. The DESCRIPTION provides a textual definition of the variable. The value after the ::=sign is the OBJECT IDENTIFIER's value which tells where the variable fits in the naming tree of Fig.4. For example, IP group is identified by {1 3 6 1 2 1 4} or {internet (1) 2 1 4}. Furthermore, related variables (objects) are collected together into groups. For example, there are IP group which describes IP packet statistics and TCP group which describes TCP traffic statistics in the MIB. Groups are further assembled into modules. The module in a managed device may have one ore more groups.

An XML document has a tree structure. It contains exactly one root element. The root element has one or more child elements. Each element may or may not have children. Therefore, the MIB can be represented easily as an XML document. An example of an XML document describing the module depicted in Fig.5, is shown in Fig.6.

This MIB module is excerpt from RFC2325 and is assigned under the Transmission group with an OID of {1 3 6 1 2 1 10 132}. It is the module for coffee vending devices. The variables : potName, potCapacity, potType and potLocation are used to describe the machine. Furthermore, the variables potOperStatus, potLevel, potMetric, potStartTime, lastStartTime and potTemperature which describe the operation of the machine are grouped into the variable potMonitor.

3.2 Data Encoding

ASN.1 transfer syntax which is used in SNMP is called Basic Encoding Rules (BER). In the following paragraphs, we will explain briefly how data are encoded by BER, according to the reference [1]. The basic encoding rules used in SNMP specify that every value transmitted, both primitive and constructed data types consists of 3 fields:

i1) The identifier (type of data).
i2) The length of the data field, in bytes.
i3) The data field.

The first field has three subfields, as shown in Fig.7. The high-order 2 bits identify tag type. The next bit tells whether the value is primitive (0) or constructed (1). The remaining 5 bits are used to encode the value of the tag in the range 0 through 30. If the tag is 31 or more, the low-order 5 bits contain 11111. Then, each identifier byte following the first one contains 7 data bits. The high-order bit is set to 0 in all but the last one.

Following the identifier field comes a field telling how many bytes the data occupy. Lengths shorter than 128 bytes are encoded in 1 byte whose left most bit is 0.

```
potCapacity OBJECT-TYPE
   SYNTAX Integer32
   MAX-ACCESS read-only
   STATUS current
   DESCRIPTION
     "The number of units of beverage supported by this device
      (regardless of its current state)."
   ::= {1 3 6 1 2 1 10 132 2}
```

Fig.3 An example of network management variable

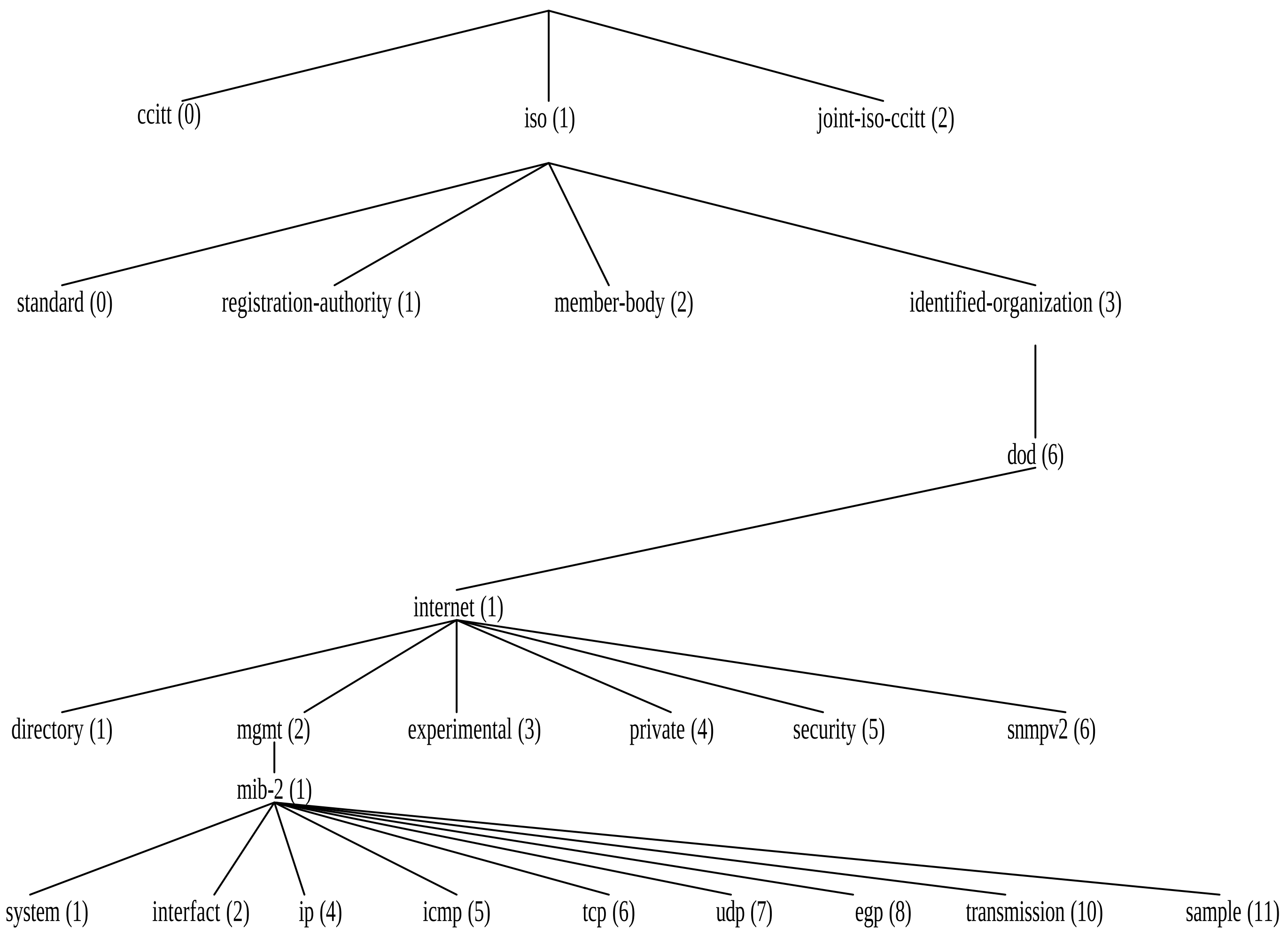


Fig.4 Part of the ASN.1 object naming tree
(From Tanenbaum, A.,S., Computer Networks

```
COFFEE-POT-MIB DEFINITIONS ::= BEGIN

coffee MODULE-IDENTITY
    LAST-UPDATED "9803231700Z"
    ORGANIZATION "Networked Appliance Management Working Group"

    CONTACT-INFO
         "       Michael Slavitch
                 Loran Technologies,
                 955 Green Valley Crescent
                 Ottawa, Ontario Canada K2A 0B6

           Tel: 613-723-7505
           Fax: 613-723-7209
         E-mail: slavitch@loran.com"
    DESCRIPTION
         "The MIB Module for coffee vending devices."
    ::= { transmission 132 }
```

```
potName OBJECT-TYPE
     SYNTAX     DisplayString (SIZE (0..255))
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "The vendor description of the pot under management"
     ::= { coffee 1 }

potCapacity OBJECT-TYPE
     SYNTAX Integer32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
       "The number of units of beverage supported by this device
        (regardless of its current state) ."
     ::= { coffee 2 }

potType OBJECT-TYPE
     SYNTAX     INTEGER {
       automatic-drip(1),
       percolator(2),
       french-press(3),
       espresso(4),
       }
     MAX-ACCESS read-write
     STATUS current
     DESCRIPTION
            "The brew type of the coffee pot."
     ::= { coffee 3 }

potLocation OBJECT-TYPE {
     SYNTAX     DisplayString (SIZE (0..255))
     MAX-ACCESS read-write
     STATUS current
     DESCRIPTION
            "The physical location of the pot in question"
     ::= { coffee 4 }

potMonitor           OBJECT IDENTIFIER ::= { coffee 6 }

potOperStatus
     SYNTAX     Integer {
                 off(1),
                 brewing(2),
                 holding(3),
                 other(4),
                 waiting(5)
                 }
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
            "The operating status of the pot in question. Note
             that this is a read-only feature. Current hardware
             prevents us from changing the port state via SNMP."
     ::= { potMonitor 1 }
```

```
potLevel OBJECT-TYPE
     SYNTAX     Integer32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
            "The number of units of coffee under management. The
             units of level are defined in potMetric below."
     ::= { potMonitor 2 }

potMetric  OBJECT-TYPE
     SYNTAX     Integer {
                espresso(1),
                demi-tasse(2),
                cup(3),
                mug(4),
                bucket(5)
                }
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
            "The vendor description of the pot under management"
     ::= { potMonitor 3 }

potStartTime OBJECT-TYPE
    SYNTAX     Integer64
    MAX-ACCESS read-write
    STATUS     current
    DESCRIPTION
           "The time in seconds since Jan 1 1970 to start the pot
            if and only if potOperStatus is waiting(5)"
    ::= { potMonitor 4 }
lastStartTime  OBJECT-TYPE
    SYNTAX     TimeInterval
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
           "The amount of time, in TimeTicks, since the coffee
           making process was initiated."
    ::= { potMonitor 5 }

potTemperature OBJECT-TYPE
     SYNTAX     Integer32
     UNITS      "degrees Centigrade"
     MAX-ACCESS read-only
     STATUS     current
     DESCRIPTION
            "The ambient temperature of the coffee within the pot"

     ::= { potMonitor 6 }

END
```

Fig.　An example of MIB module

```
<MIB VER     NO    >
 <MODULE> COFFEE < MODULE>
 <LAST-UPDATED>                  < LAST-UPDATED>
 <ORGANIZATION> NETWORKED APPLIANCE MANAGEMENT WORKING GROUP
  < ORGANIZATION>
 <OID> { TRANSMISSION      } < OID>
 <OBJECT>
  <TYPE> POTNAME </TYPE>
  <SYNTAX> STRING </SYNTAX>
  <MAX-ACCESS> READ-ONLY </MAX-ACCESS>
  <STATUS> CURRENT </STATUS>
  <DESCRIPTION> THE VENDOR DESCRIPTION OF THE POT UNDER MANAGEMENT
   </DESCRIPTION>
  <OID> { COFFEE 1 } </OID>
 </OBJECT>
<OBJECT>
  <TYPE> POTCAPACITY </TYPE>
  <SYNTAX> INTEGER </SYNTAX>
  <MAX-ACCESS> READ-ONLY </MAX-ACCESS>
  <STATUS> CURRENT </STATUS>
  <DESCRIPTION> THE NUMBER OF UNITS OF BEVERAGE SUPPORTED BY THIS DEVICE
   </DESCRIPTION>
  <OID> { COFFEE 2 } </OID>
 </OBJECT>

 <OBJECT>
   <TYPE> POTTYPE </TYPE>
   <SYNTAX> INTEGER
    <SUB>
     <VALUE> 1
      <IS> AUTOMATIC – DRIP</IS>
     </VALUE>
     <VALUE> 2
      <IS> PERCOLATOR </IS>
      </VALUE>
      <VALUE> 3
       <IS> FRENCH-PRESS </IS>
      </VALUE>
      <VALUE> 4
       <IS> ESPRESSO </IS>
      </VALUE>
     </SUB>
    </SYNTAX>
    <MAX-ACCESS> READ-WRITE </MAX-ACCESS>
    <STATUS> CURRENT </STATUS>
    <DESCRIPTION> THE BREW TYPE OF THE COFFEE POT </DESCRIPTION>
    <OID> { COFFEE 3 } </OID>
   </OBJECT>

   <OBJECT>
    <TYPE> POTLOCATION </TYPE>
    <SYNTAX> STRING </SYNTAX>
    <MAX-ACCESS> READ-WRITE </MAX-ACCESS>
    <STATUS> CURRENT </STATUS>
    <DESCRIPTION> THE PHYSICAL LOCATION OF THE POT IN QUESTION </DESCRIPTION>
    <OID> { COFFEE 4 } </OID>
   </OBJECT>
```

```
</GROUP>
 POTMONITOR
 <OID> { COFFEE 6 } </OID>

 <OBJECT>
   <TYPE> POT-OPER-STATUS </TYPE>
   <SYNTAX> INTEGER
     <SUB>
      <VALUE> 1
       <IS> OFF </IS>
      </VALUE>
      <VALUE> 2
       <IS> BREWING </IS>
      </VALUE>
      <VALUE> 3
       <IS> HOLDING </IS>
      </VALUE>
      <VALUE> 4
       <IS> OTHER </IS>
      </VALUE>
      <VALUE> 5
       <IS> WAITING </IS>
      </VALUE>
     </SUB>
   </SYNTAX>
   <MAX-ACCESS> READ-ONLY </MAX-ACCESS>
   <STATUS> CURRENT </STATUS>
   <DESCRIPTION> THE OPERATING STATUS OF THE POT IN QUESTION </DESCRIPTION>
   <OID> { POTMONITOR 1 } </OID>
 </OBJECT>

 <OBJECT>
   <TYPE> POTLEVEL </TYPE>
   <SYNTAX> INTEGER </SYNTAX>
   <MAX-ACCESS> READ-ONLY </MAX-ACCESS>
   <STATUS> CURRENT </STATUS>
   <DESCRIPTION> THE NUMBER OF UNIT OF COFFEE UNDER MANAGEMENT </DESCRIPTION>
   <OID> { POTMONITOR 2 } </OID>
 </OBJECT>

 <OBJECT>
   <TYPE> POTMETRIC </TYPE>
   <SYNTAX> INTEGER
     <SUB>
       <VALUE> 1
          <IS> ESPRESSO </IS>
       </VALUE>
       <VALUE> 2
         <IS> DEMI-TASSE </IS>
       </VALUE>
       <VALUE> 3
         <IS> CUP </IS>
       </VALUE>
       <VALUE> 4
         <IS> MUG </IS>
      </VALUE>
      <VALUE> 5
        <IS> BUCKET </IS>
      </VALUE>
      </SUB>
     </SYNTAX>
   <MAX-ACCESS> READ-ONLY </MAX-ACCESS>
   <STATUS> CURRENT </STATUS>
   <DESCRIPTION> THE VENDOR DESCRIPTION OF THE POT UNDER MANAGEMENT
     </DESCRIPTION>
   <OID> { POTMONITOR 3 } </OID>
  </OBJECT>

  <OBJECT>
   <TYPE> POT-START-TIME </TYPE>
   <SYNTAX> INTEGER </SYNTAX>
   <MAX-ACCESS> READ-WRITE </MAX-ACCESS>
   <STATUS> CURRENT </STATUS>
   <DESCRIPTION> THE TIME IN SECONDS SINCE JAN 1 1970 TO START THE POT </DESCRIPTION>
   <OID> { POTMONITOR 4 } </OID>
  </OBJECT>

  <OBJECT>
   <TYPE> LAST-START-TIME </TYPE>
   <SYNTAX> INTEGER </SYNTAX>
   <MAX-ACCESS> READ-ONLY </MAX-ACCESS>
   <STATUS> CURRENT </STATUS>
   <DESCRIPTION> THE AMOUNT OF TIME IN TIMETICKS SINCE THE COFFEE MAKING PROCESS
     WAS INITIATED </DESCRIPTION>
   <OID> { POTMONITOR 5 } </OID>
  </OBJECT>

  <OBJECT>
   <TYPE> POT-TEMPERATURE </TYPE>
   <SYNTAX> INTEGER </SYNTAX>
   <UNIT> DEGREE CENTIGRADE </UNIT>
   <MAX-ACCESS> READ-ONLY </MAX-ACCESS>
   <STATUS> CURRENT </STATUS>
 <DESCRIPTION> THE AMBIENT TEMPERATURE OF THE COFFEE WITHIN THE POT
   </DESCRIPTION>
  <OID> { POTMONITOR 6 } </OID>
 </OBJECT>
</GROUP>
</MIB>
```

Fig.6 An XML document describing the MIB module of Fig.5

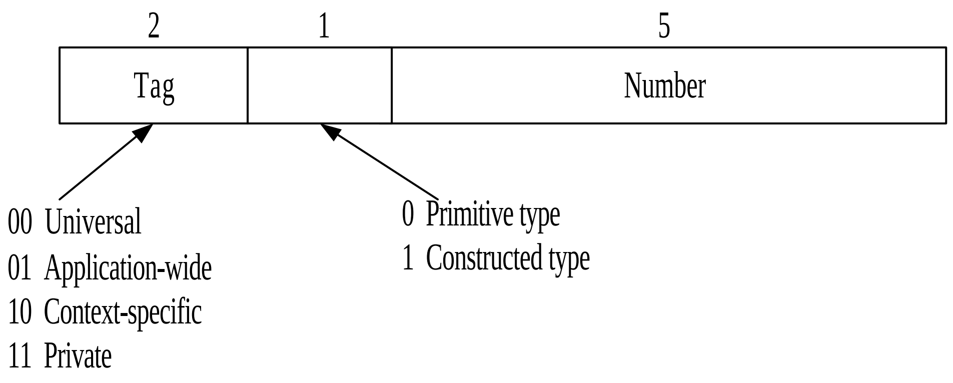
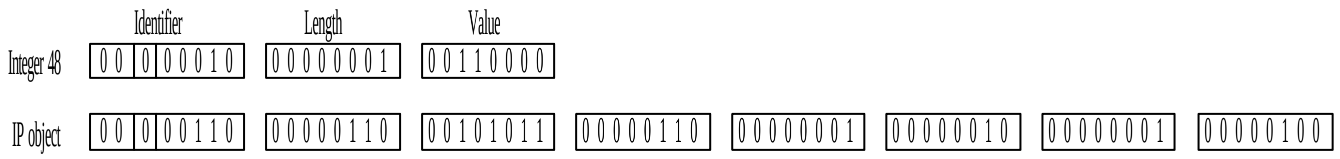


Fig.7 The first byte of the identifier field

Those that are longer use multiple bytes, with first byte containing a 1 in the high – order bit and the length field in the low-order 7 bits. For example, if the data length is 1000 bytes, the first byte contains 130 to indicate a two byte length field follows. Then come two bytes whose value is 1000.

The encoding of the data filed depends on the type of data present. For example, integers are encoded in two's complement. An OBJECT IDENTIFIER is encoded as the sequence of integers it represents. For example, IP object is { 1 3 6 1 2 1 4 }. Since the first number is always 0, 1, or 2, and the second is less than 40, the first two number, a and b are encoded as 1 byte having the value 40 a+ b. For the next numbers, each is encoded in the same way as the length field. Fig 8 (a) shows an example of encoding some data values.

In the following sentences we describe how the transferred data is sent in the form of XML. Since only some types shown in Fig 9 are allowed in SNMP. This data types can be simply encoded. The length field is not needed because the opening tags and closing tags are used in XML. The data is simply sent as character string. An example of transferring data using XML document is shown in Fig 8 (b).



(a)

Integer 48	<CODE> 0 </CODE>
	<VALUE> 48 </VALUE>
IP object	<CODE> 10 </CODE>
	<VALUE> { 1 3 6 1 2 1 4 } </VALUE>

(b)

Fig.8 (a) Examples of encoding transferred data by using ASN.1
(b) Examples of transferred data using XML

Data Type	Description	CODE
INTEGER	Integers in the range of -2^{31} to $2^{31} - 1$.	0
Uinteger32	Integers in the range of 0 to $2^{31} - 1$.	1
Counter32	A nonnegative integer that may be incremented modulo 2^{32}.	2
Counter64	A nonnegative integer that may be incremented modulo 2^{64}.	3
Gauge32	A nonnegative integer that may increase or decrease, but shall not exceed a maximum value. The maximum value can not be greater that $2^{32} - 1$.	4
Time Ticks	A nonnegative integer that represents the time, modulo 2^{32}, in hundredths of a second.	5
OCTET STRING	Octet strings for arbitrary binary or textual data: may be limited to 255 octets.	6
Ip Address	A 32–bit internet address.	7
Opaque	An arbitrary bit field.	8
BIT STRING	An enumeration of named bits.	9
OBJECT IDENTIFIER	Administratively assigned name to object or other standardized element.	10

(a) (b)

Fig.9 (a) Allowable Data Types in SNMPv2
(From Stallings, W. ; Data and Computer Communications)[3]
(b) Encoded value for the Data Types

4. Data Exchange

Data exchange between a management station and managed nodes is done by using SNMP protocol Fig shows some massages which can be sent according to the protocol. Usually, the manager send a request asking for information or commanding the agent to update its state. The agent replies with the requested information or confirms that it has updated its state as requested. Fig.11 (a) shows an example of request message sent by the management station. Fig.11 (b) shows the response message represented as character strings. Furthermore, Fig.11 (c) and (d) shows the message really sent as ASN.1 and encoded data, respectively. As shown in Fig.11 (c), the response message is sent as a sequence of several data types. Further, the information of each object is sent as a sequence of the OID, the datatype and the value of the object. As SNMP protocol is a text based one, we can use XML documents to present the exchanged messages. Fig.12 shows an example of XML documents sent in the place of the messages in Fig.11.

Message	Description
Get–request	**Requests the value of one or more variables**
Get–next–request	**Requests the variable following this one**
Get–bulk–request	**Fetches a large table**
Set–request	**Updates one or more variables**

Fig.10 Some SNMP message types

```
SNMP: ----- Simple Network Management Protocol -------
SNMP:
SNMP: Version = 0
SNMP: Community = boulder
SNMP: Command = Get Request
SNMP: Requets ID = 0
SNMP: Error status = 0 (No error)
SNMP: Error index = 0
SNMP:
SNMP: Object = { 1 3 6 1 2 1 1 3 0 }
SNMP: Value = NULL
SNMP:
SNMP: Object = { 1 3 6 1 2 1 1 1 0 }
SNMP: Value = NULL
SNMP:
SNMP: Object = { 1 3 6 1 2 1 1 2 0 }
SNMP: Value = NULL
SNMP:
SNMP: Object = { 1 3 6 1 2 1 1 3 0 }
SNMP:
SNMP: Object = { 1 3 6 1 2 1 1 4 0 }
SNMP: Value = NULL
SNMP:
SNMP: Object = { 1 3 6 1 2 1 1 5 0 }
SNMP: Value = NULL
SNMP:
SNMP: Object = { 1 3 6 1 2 1 1 6 0 }
SNMP: Value = NULL
SNMP:
SNMP: Object = { 1 3 6 1 2 1 1 7 0 }
SNMP: Value = NULL
SNMP:
```

(a)

```
SNMP: ----- Simple Network Management Protocol -------
SNMP:
SNMP: Version = 0
SNMP: Community = boulder
SNMP: Command = Get Response
SNMP: Requets ID = 0
SNMP: Error status = 0 (No error)
SNMP: Error index = 0
SNMP:
SNMP: Object = { 1 3 6 1 2 1 1 3 0 }
SNMP: Value = 263621778 hundredths of second
SNMP:
SNMP: Object = { 1 3 6 1 2 1 1 1 0 }
SNMP: Value = Portable I80386 C Gateway BOULDER.ORG S/N XXX V12.0
SNMP:
SNMP: Object = { 1 3 6 1 2 1 1 2 0 }
SNMP: Value = { 1 3 6 1 4 1 1 1 1 41 }
SNMP:
SNMP: Object = { 1 3 6 1 2 1 1 3 0 }
SNMP: Value = 263621778 hundredths of second
SNMP:
SNMP: Object = { 1 3 6 1 2 1 1 4 0 }
SNMP: Value =
SNMP:
SNMP: Object = { 1 3 6 1 2 1 1 5 0 }
SNMP: Value = BOULDER.ORG
SNMP:
SNMP: Object = { 1 3 6 1 2 1 1 6 0 }
SNMP: Value =
SNMP:
SNMP: Object = { 1 3 6 1 2 1 1 7 0 }
SNMP: Value = 72
SNMP:
```

(b)

```
SNMP: 1.1 SEQUENCE [of], Length=235
SNMP: 2.1 INTEGER, Length=1, Value = "0"
SNMP: 2.2 OCTET STRING, Length=7, Value = "boulder"
SNMP: 2.3 Context-Specific Constructed [2], Length=220
SNMP: 3.1 INTEGER, Length=1, Value = "0"
SNMP: 3.2 INTEGER, Length=1, Value = "0"
SNMP: 3.3 INTEGER, Length=1, Value = "0"
SNMP: 3.4 SEQUENCE [of], Length=208
SNMP: 4.1 SEQUENCE [of], Length=16
SNMP: 5.1 OBJECT IDENTIFIER, Length=8, Value = "{1 3 6 1 2 1 1 3 0}"
SNMP: 5.2 Application Primitive [3], Length=4, Data = "<0FB68C92>"
SNMP: 4.2 SEQUENCE [of], Length=74
SNMP: 5.1 OBJECT IDENTIFIER, Length=8, Value = "{1 3 6 1 2 1 1 1 0}"
SNMP: 5.2 OCTET STRING, Length=62, Value = "Portable I80386 C Gateway XXX.XXX.XXXX.XXX..."
SNMP: 4.3 SEQUENCE [of], Length=21
SNMP: 5.1 OBJECT IDENTIFIER, Length=8, Value = "{1 3 6 1 2 1 1 2 0}"
SNMP: 5.2 OBJECT IDENTIFIER, Length=9, Value = "{1 3 6 1 4 1 1 1 1 41}"
SNMP: 4.4 SEQUENCE [of], Length=16
SNMP: 5.1 OBJECT IDENTIFIER, Length=8, Value = "{1 3 6 1 2 1 1 3 0}"
SNMP: 5.2 Application Primitive [3], Length=4, Data = "<0FB68C92>"
SNMP: 4.5 SEQUENCE [of], Length=12
SNMP: 5.1 OBJECT IDENTIFIER, Length=8, Value = "{1 3 6 1 2 1 1 4 0}"
SNMP: 5.2 OCTET STRING, Length=0, Value = ""
SNMP: 4.6 SEQUENCE [of], Length=28
SNMP: 5.1 OBJECT IDENTIFIER, Length=8, Value = "{1 3 6 1 2 1 1 5 0}"
SNMP: 5.2 OCTET STRING, Length=16, Value = " XXX.XXX.XXXX.XXX"
SNMP: 4.7 SEQUENCE [of], Length=12
SNMP: 5.1 OBJECT IDENTIFIER, Length=8, Value = "{1 3 6 1 2 1 1 6 0}"
```

```
SNMP: 5.2 OCTET STRING, Length=0, Value = ""
SNMP: 4.8 SEQUENCE [of], Length=13
SNMP: 5.1 OBJECT IDENTIFIER, Length=8, Value = "{1 3 6 1 2 1 1 7 0}"
SNMP: 3.3 INTEGER, Length=1, Value = "72"
SNMP:
```

(c)

```
ADDR   HEX                                                   ASCII
0000   08 00 20 09 00 C8 AA 00 04 00 44 86 08 00 45 00              ..        .......D...E.
0010   01 0A 81 20 00 00 3B 11 73 77 84 A3 01 01 84 A3              ...       ..;.sw......
0020   80 04 00 A1 0D 20 00 F6 C6 62 30 81 EB 02 01 00       ....        ...b0.....
0030   04 07 XX XX XX XX XX XX XX A2 81 DC 02 01 00 02              . .XXXXXXX .......
0040   01 00 02 01 00 30 81 D0 30 10 06 08 2B 06 01                 . . . . .0 . . 0 ...+...
0050   01 01 03 00 43 04 0F B6 8C 92 30 4A 06 08 2B 06       . . . .C. . . . .0J..+.
0060   01 02 01 01 01 00 04 3E 50 6F 72 74 61 62 6C 65              . . . . . ..>Portable
0070   20 49 38 30 33 38 36 20 43 20 47 61 74 65 77 61              I80386 C Gatewa
0080   79 20  XX XX XX XX XX XX XX  XX XX XX XX XX XX XX                    y XXX.XXX.XXXX.X
0090   XX XX 20 53 2F 4E 20 33 33 33 20 56 31 32 2E 30       XX S/N 333 V12.0
00A0   20 20 5B 20 20 5D 30 15 06 08 2B 06 01 02 01 01       [           ]0...+.....
00B0   02 00 06 09 2B 06 01 04 01 01 01 01 29 30 10 06              . . . .+ . . . . . . .)0..
00C0   08 2B 06 01 02 01 01 03 00 43 04 00 04 00 30 1C 06    . + . . . . . . .C. . . . .0
00D0   0C 06 08 2B 06 01 02 01 01 04 00 04 00 30 1C 06       . . .+ . . . . . . . . .0..
00E0   08 2B 06 01 02 01 01 05 00 04 10 XX XX XX XX XX              .+. . . . . . . . .XXX.X
00F0   XX XX XX XX XX XX XX XX XX XX XX 30 0C 06 08 2B              XX.XXXX.XXX0...+
0100   06 10 02 01 01 06 00 04 00 30 0D 06 08 2B 06 01       . . . . . . . . .0. . .+. .
0110   02 01 01 07 00 02 01 48                               . . . . . . .  H
```

(d)

Fig.11 (a) An example of request message
(b) An example of response message
(c) Response message sent in the form of ASN.1
(d) Hexa values of the encoded message of Fig.(c)

```
<SNMP>
  <Version> 0  </Version>
  <Community> boulder </Community>
  <Command> Get Request </Command>
  <RequetsID> 0 </RequetsID>
  <ErrorStatus> 0 </ErrorStatus>
  <ErrorIndex> 0 </ErrorIndex>
  <DATA>
    <Object> { 1 3 6 1 2 1 1 3 0 }
      <Value></Value>
    </Object>
    <Object>  { 1 3 6 1 2 1 1 1 0 }
      <Value></Value>
    </Object>
    <Object>  { 1 3 6 1 2 1 1 2 0 }
      <Value></Value>
    </Object>
    <Object>  { 1 3 6 1 2 1 1 3 0 }
      <Value></Value>
    </Object>
    <Object>  { 1 3 6 1 2 1 1 4 0 }
      <Value></Value>
    </Object>
    <Object> { 1 3 6 1 2 1 1 5 0 }
      <Value></Value>
    </Object>
    <Object> { 1 3 6 1 2 1 1 6 0 }
      <Value></Value>
    </Object>
    <Object>  { 1 3 6 1 2 1 1 7 0 }
      <Value></Value>
    </Object>
  </DATA>
</SNMP>
```

(a)

```
<SNMP>
  <Version> 1  </Version>
  <Community> boulder </Community>
  <Command> Get Response </Command>
  <RequetsID> 0 </RequetsID>
  <ErrorStatus> 0 </ErrorStatus>
  <ErrorIndex> 0 </ErrorIndex>
  <DATA>
    <Object> { 1 3 6 1 2 1 1 3 0 }
      <Code> 5 </Code>
      <Value> 263621778 </Value>
    </Object>
    <Object>  { 1 3 6 1 2 1 1 1 0 }
      <Code> 6 </Code>
      <Value> Portable I80386 C Gateway BOULDER.ORG S/N XXX V12.0 </Value>
    </Object>
    <Object>  { 1 3 6 1 2 1 1 2 0 }
      <Code> 10 </Code>
      <Value> { 1 3 6 1 4 1 1 1 1 41 } </Value>
    </Object>
    <Object>  { 1 3 6 1 2 1 1 3 0 }
     <Code> 5 </Code>
     <Value> 263621778 </Value>
    </Object>
    <Object>  { 1 3 6 1 2 1 1 4 0 }
      <Code> 6 </Code>
      <Value></Value>
    </Object>
    <Object> { 1 3 6 1 2 1 1 5 0 }
      <Code> 6 </Code>
      <Value> BOULDER.ORG </Value>
    </Object>
    <Object> { 1 3 6 1 2 1 1 6 0 }
      <Code> 6 </Code>
      <Value></Value>
    </Object>
    <Object>  { 1 3 6 1 2 1 1 7 0 }
      <Code> 0 </Code>
      <Value> 72 </Value>
    </Object>
  </DATA>
</SNMP>
```

(b)

Fig.12 (a) An example of request message in the form of XML
(b) An example of response message in the form of XML

5. An example of management system

In the previous sections, we have described the idea how XML is used in SNMP model. In this section, we will show an example of a system which implements that idea. In the system, we use a PC with the browser as a management station. The managed node is a PC working as a web server. Each web server has a log file which contains the information about the web access. Fig.13 depicts an example of the log file. We have constructed a MIB module called WebAcess and place it under the Transmission group with an OID of {1 3 6 1 2 1 10 175}. Fig.14 shows the module.

After the authentication process, the manager sends a request message as XML document depicted in Fig.15 (a). When receiving the message, the agent parses the XML message and extracts the request by using the DOM (Document Object Model) technique. After checking the request with the MIB, the agent access the data from the log file. The response message is then formed as an XML document and sent back to the manager. Fig.15 (b) depicts the XML response.

```
#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2002-05-31 03:11:55
#Fields: date time c-ip cs-username s-ip s-port cs-method cs-uri-stem cs-uri-query sc-status cs(User-Agent)
2002-05-31 03:11:55 127.0.0.1 - 127.0.0.1 80 GET /iisstart.asp - 302
Mozilla/4.0+(compatible;+MSIE+5.0;+Windows+2000)+Opera+6.01++[en]
2002-05-31 03:11:55 127.0.0.1 - 127.0.0.1 80 GET /localstart.asp - 401
Mozilla/4.0+(compatible;+MSIE+5.0;+Windows+2000)+Opera+6.01++[en]
2002-05-31 03:12:06 127.0.0.1 aaa 127.0.0.1 80 GET /localstart.asp - 200
Mozilla/4.0+(compatible;+MSIE+5.0;+Windows+2000)+Opera+6.01++[en]
```

Fig.13 An example of web-access log file

```
<MIB VER="1" NO="1">
  <MODULE> WEBACCESS </MODULE>
  <LAST-UPDATED> 9803231700 </LAST-UPDATED>
  <ORGANIZATION> NIDA, APPLIED STATISTIC </ORGANIZATION>
  <OID> { TRANSMISSION 175 } </OID>

  <OBJECT>
    <TYPE> DATE-TIME </TYPE>
    <SYNTAX> TimeTicks </SYNTAX>
    <MAX-ACCESS> READ-ONLY </MAX-ACCESS>
    <STATUS> CURRENT </STATUS>
    <DESCRIPTION> DATE-TIME OF WEB ACCESS </DESCRIPTION>
    <OID> { WEBACCESS 1 } </OID>
  </OBJECT>

  <OBJECT>
    <TYPE> C-IP </TYPE>
    <SYNTAX> IpAddress </SYNTAX>
    <MAX-ACCESS> READ-ONLY </MAX-ACCESS>
    <STATUS> CURRENT </STATUS>
    <DESCRIPTION> IP ADDRESS OF CLIENT </DESCRIPTION>
    <OID> { WEBACCESS 2 } </OID>
  </OBJECT>

  <OBJECT>
    <TYPE> S-IP </TYPE>
    <SYNTAX> IpAddress </SYNTAX>
    <MAX-ACCESS> READ-ONLY </MAX-ACCESS>
    <STATUS> CURRENT </STATUS>
    <DESCRIPTION> IP ADDRESS OF WEB SERVER </DESCRIPTION>
    <OID> { WEBACCESS 3 } </OID>
  </OBJECT>

  <OBJECT>
    <TYPE> S-PORT </TYPE>
    <SYNTAX> INTEGER </SYNTAX>
    <MAX-ACCESS> READ-ONLY </MAX-ACCESS>
    <STATUS> CURRENT </STATUS>
    <DESCRIPTION> PORT ON WEB SERVER </DESCRIPTION>
    <OID> { WEBACCESS 4 } </OID>
  </OBJECT>

  <OBJECT>
```

```
    <TYPE> CS-METHOD </TYPE>
    <SYNTAX> OCTET STRING </SYNTAX>
    <MAX-ACCESS> READ-ONLY </MAX-ACCESS>
    <STATUS> CURRENT </STATUS>
    <DESCRIPTION> METHOD THAT CLIENT REQUEST TO WEB SERVER </DESCRIPTION>
    <OID> { WEBACCESS 5 } </OID>
  </OBJECT>

  <OBJECT>
    <TYPE> CS-URI-STREM </TYPE>
    <SYNTAX> OCTET STRING </SYNTAX>
    <MAX-ACCESS> READ-ONLY </MAX-ACCESS>
    <STATUS> CURRENT </STATUS>
    <DESCRIPTION> PAGE THAT CLIENT REQUEST </DESCRIPTION>
    <OID> { WEBACCESS 6 } </OID>
  </OBJECT>

  <OBJECT>
    <TYPE> CS-STATUS </TYPE>
    <SYNTAX> INTEGER </SYNTAX>
    <MAX-ACCESS> READ-ONLY </MAX-ACCESS>
    <STATUS> CURRENT </STATUS>
    <DESCRIPTION> CODE NUMBER OF RESPONSE </DESCRIPTION>
    <OID> { WEBACCESS 7 } </OID>
  </OBJECT>

  <OBJECT>
    <TYPE> CS </TYPE>
    <SYNTAX> OCTET STRING </SYNTAX>
    <MAX-ACCESS> READ-ONLY </MAX-ACCESS>
    <STATUS> CURRENT </STATUS>
    <DESCRIPTION> CLIENT SYSTEM SUCH AS OS, BROWSER </DESCRIPTION>
    <OID> { WEBACCESS 8 } </OID>
  </OBJECT>

  <OBJECT>
    <TYPE> SUM </TYPE>
    <SYNTAX> INTEGER </SYNTAX>
    <MAX-ACCESS> READ-ONLY </MAX-ACCESS>
    <STATUS> CURRENT </STATUS>
    <DESCRIPTION> NUMBER OF WEBACCESS </DESCRIPTION>
    <OID> { WEBACCESS 9 } </OID>
  </OBJECT>

</MIB>
```

Fig.14 An example of the web-access module described by using XML

```
<SNMP>
        <Version> 1 </Version>
        <Community> public </Community>
        <Command> Get Request </Command>
        <RequetsID> 0 </RequetsID>
        <ErrorStatus> 0 </ErrorStatus>
        <ErrorIndex> 0 </ErrorIndex>
        <Data>
                <Object> { 1 3 6 1 2 1 10 175 1 }
                        <Value></Value>
                </Object>
                <Object> { 1 3 6 1 2 1 10 175 2 }
                        <Value></Value>
                </Object>
                <Object> { 1 3 6 1 2 1 10 175 3 }
                        <Value></Value>
                </Object>
        </Data>
</SNMP>

<SNMP>
        <Version> 1 </Version>
        <Community> public </Community>
        <Command> Get Response </Command>
        <RequetsID> 0 </RequetsID>
        <ErrorStatus> 0 </ErrorStatus>
        <ErrorIndex> 0 </ErrorIndex>
        <Data>
                <Object> { 1 3 6 1 2 1 10 175 1 }
                    <Code> 5 </Code>
                      <Value> 1297152600 </Value>
                </Object>
                <Object> { 1 3 6 1 2 1 10 175 2 }
                    <Code> 7 </Code>
                      <Value> 127.0.0.1 </Value>
                </Object>
                <Object> { 1 3 6 1 2 1 10 175 3 }
                    <Code> 7 </Code>
                      <Value> 127.0.0.1 </Value>
                </Object>
        </Data>
</SNMP>
```

Fig.15 An example of request-response message in the form of XML

6. Conclusions

This paper reports an application of XML on SNMP. In the paper, XML is used to describe data in the MIB in stead of ASN.1 As XML document has a tree structure, it can replace ASN. 1 easily. Furthermore, the management information can be described precisely by using XML tags. The data exchanged between the management station and the managed nodes is in the form of XML. Because XML documents are text based, the data-exchange process is simple. The data received at the management station can be easily presented to the manager. They can also be used in further processing. By using XML, SMNP can be implemented as web application easily and effectively.

References

[1] Andrew S. Tanenbaum ; Computer Networks, 3rd edition, Prentice – Hall, 1996,

[2] Harvey M. Deitel, Paul J. Deitel, Tem R. Nieto, Ted Lin and Praveen Sadhu ; XML How to Program , Prentice – Hall, 2001.

[3] William Stallings ; Data and Comp uter Communications, 6th edition , Prentice Hall, 2000.

[4] Mark A. Miller ; Managing Internetworks with SNMP, 3rd edition, M&T Books, 1999.

[5] Douglas E. Comer and David L. Stevens ; Internetworking with TCP/IP, 3rd edition, Prentice-Hall, 2001.