Managing the Apache HTTP server with SNMP

Author: Harrie Hazewinkel
Date: March 12, 2001

Abstract

The Apache HTTP server is the core component in any organizations that do business on the Internet. Many organizations are introducing QoS management as essential ingredient for web services. This talk will show what it takes to implement SNMP management, how you use it, and actual examples of configurations, scripts and management applications. It will introduce the various relevant protocol mechanisms which enables you to manage your network and the MIB modules which define your management information. The talk will provide a basic knowledge of how to read and use the MIB modules.

1.0 Introduction

With the growth of the Internet the networks became complex and difficult to manage. In particular the World Wide Web application is becoming important and its usability depends upon the performance realized by the WWW servers. Many organization have the log files as their only mean of monitoring. As the amount of WWW servers in organizations grow, the management of these servers becomes mostly a problem. To many systems and to many log files that must be processed real-time to provide the current health of all your WWW servers. By applying the Internet Management to in the application, one can manage near real-time the Apache HTTP server. Many other systems, such as routers, have already standard Internet management functionality known as the Simple Network Management Protocol (SNMP). SNMP is based upon open standards and has al;ready a huge deployment over the Internet. Many systems, such as routers and hubs, have SNMP and the systems are able to be monitored with standard management framework applications. These framework applications are easy extensable and that allows you to leverage your investments in these application frameworks.
This paper describes how an Apache HTTP server can be managed with SNMP. It provdes as quick overview of the SNMP framework and is followed by summaries of a number of MIB modules that are usefull for management of the Apache HTTP server. Two MIB modules, the WWW-MIB module and the NETWORK-SERVICES-MIB module, are described in more detail on how these MIB modules could be of assistance to manage your Apache HTTP server.

2.0 The SNMP framework

The basic structure of SNMP comprises out of 2 components. The managers which are network management stations and the agents which are the managed nodes. An SNMP manager is an application with an SNMP entity that may invoke SNMP requests on SNMP agents and receives notifications from SNMP agents. An SNMP manager is the application that may interact with an operator by for instance issues alarms or it may take automated actions based upon an observed status. The SNMP agents (typically many) are the managed nodes that provide access to management information. The SNMP agent responds to SNMP queries and may initiate notifications to a management application. The management information is represented in a MIB that is composed of various modules of management information and represents the managed device/component.

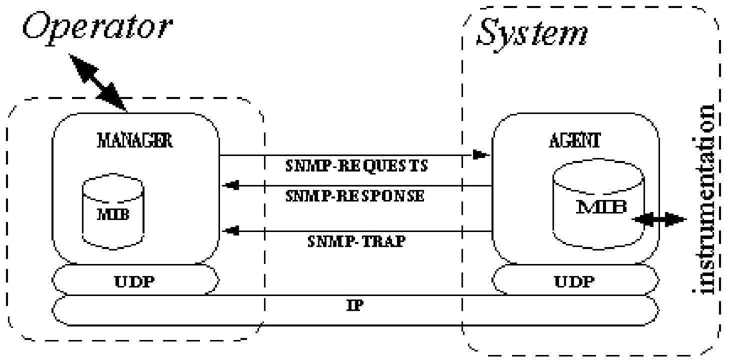

Figure 1: SNMP framework

In figure 1 is depicted the SNMP framework with the manager and agent communication. The 'Operator' perceives a coherent view of the managed systems and their coherence via the SNMP manager. Sometimes also called a management application. The SNMP manager gets access to the managed instrumentation by invoking requests on a virtual information store, the Managed Information Base (MIB), that resides in an SNMP agent. The MIB is a collection of managed objects that represent the managed instrumentation. I.e. the MIB could be seen as some kind of switchboard for the managed system.

2.1 SNMP versions

- SNMP version 1 (SNMPv1) is the very first version of SNMP that was derived from and enhanced the Simple Gateway Management Protocol (SMGP). This version included not only protocol operations, but also a data definition language, Structure of Management Information (SMIv1) and had a community based security. A community was a 'secret identifier' that authenticated access to the management data in an SNMP agent.
- SNMP version 2 Community based (SNMPv2c) was the first attempt to improve the known operational problems and should have added security. However, SNMPv2c as currently defined has still the community based security as known in SNMPv1, but it provided additional data types for the SMI (now known as SMIv2) and additional protocol operations to optimize network traffic.
- SNMP version 3 (SNMPv3) is the newest version and provides user based security and view based access control. The user based security is based upon the concept of users and to authenticate they need to provide a password. This is similar as the commonly known login procedures for computers. The view based access control is nothing more then a particular view an associated SNMPv3 user can have of the Management Information Base (MIB). In simple terms, what is the operator able to see and what is the operator not able to see.

3.0 The management Information Base

The SNMP operations conveyed in the SNMP requests are acting upon the Management Information Base (MIB). The MIB is a virtual information store that consists of a set of managed objects. These managed objects represent the instrumentation that is represented by the managed object. Theoretically, the operations act upon the managed objects, but in practice they act upon the instrumentation. Typically, a MIB composed of MIB modules that each define the managed objects and such a MIB module is focussed on a particular type of managed objects.

3.1 Structure of Management Information

The MIB modules that comprise the MIB are defined in a special data definition language. The language, termed Structure of Management Information (SMI), is an adapted subset of ASN.1 (Abstract Syntax Notation 1). The SMI specifies constructs to define to kinds of managed objects, scalar objects and columnar objects. Scalar objects are single valued objects that are only instantiated once in a system. The columnar objects are ordered within a a conceptual row that resides in a conceptual table. The columnar objects can occur more than once in the system, but each instance is uniquely indexed. The SMI provides mechanisms to define these objects as well other definition handles such as, a module identification construct, conformance statements.
The language also defines data types that can be used for the managed objects.

3.2 The naming tree

The MIB is built around a naming tree comprised of object identifiers. These object identifiers (OID) are a sequence of numbers that represent the position in the tree. The tree starts a 'root'-node after which the naming starts. The leafs of the tree are the managed objects that instantiate the instrumentation. However, the naming tree shows an hierarchy a MIB is not object oriented (OO) in which higher positioned objects inherit from the lower objects. Also each node in the tree has mostly not only a number, but have also a name for human readability.

Although, there many SNMP protocol operations, the operations for searching managed objects in the MIB tree is limited to two;

1. the **exact search** that requires from the SNMP protocol operation to have the exact match of the OID to operate on a managed object,
2. the **next search** that specifies that the operation acts upon the lexical graphical next available managed object.

A MIB tree is built out of numbered nodes. A node is identified by a sequence of numbers starting always from the top node (root)
The figure below depicts an example MIB tree. The all nodes have a number as well a human readable name. The OID of the **sysUpTime** managed object is **1.3.6.1.2.1.1.3**. The means when the value of the instance of the sysUpTime is retrieved via the exact search the operation is done on **sysUpTime** or **1.3.6.1.2.1.1.3.0**. However, for a next search would only require **sysUpTime** (or **1.3.6.1.2.1.1.3**), since the lexical graphical next object would be **sysUpTime.0** (or **1.3.6.1.2.1.1.3.0**). The figure below also contains the lexical graphical order of the managed objects. This is depicted by the arrows that start at **sysDescr.0** which is the first lexical graphical object and going via the managed objects (instances) and finally traverse through the **sysORTable**. In the figure the exception '**endOfMIBView**' indicates that there are no next available managed object of the last s**ysORUpTime.3**.

.

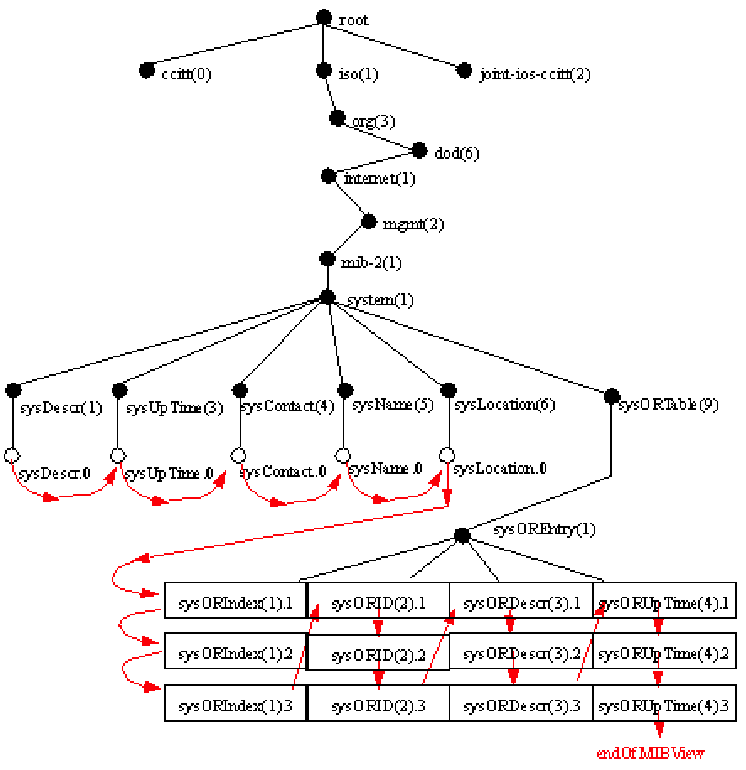

figure 1: MIB tree

3.3 Enterprise specific MIB modules

Many MIB modules are already defined and standardized. However, these are always a common set of managed objects. Mostly enterprises add specific functionality and for instance configuration directives to implementations of standards. The SMI allows therefore also to define enterprise specific MIB modules. An enterprise specific MIB module must be made under a 'privately owned' branch in the MIB tree.

4.0 MIB modules

The core component of an SNMP agent is the Management Information Base (MIB). A MIB is composed of various MIB modules that represent the managed device. The objective of this section is to provide an overview of IETF standardized MIB modules that are potentially useful for management of an Apache HTTP server. This section provides only a summary of those MIB modules. The two most applicable application MIB modules are each described in a separate section (WWW-MIB, NETWORK-SERVICES-MIB).

4.1 SNMP framework MIB modules

The network management framework does require some basic managed objects (defined in SNMPv2-MIB) for its system info and for the SNMP protocol. Additional to this some MIB modules are specifically made to support the SNMPv3 security framework. This section provides the list of the SNMP framework MIB modules.

- SNMPv2-MIB

  A standard required MIB for SNMP agents that provides managed objects for the system, snmp statistics and well known notifications. This MIB module is defined in RFC 1907.
- SNMP-FRAMEWORK-MIB

  The SNMP architecture MIB providing generic managed objects and definitions for SNMPv3. This MIB module is defined in RFC 2571.
- SNMP-MPD-MIB

  This MIB module contains management information definitions for the Message Processing and Dispatching for SNMPv3. This MIB module is defined in RFC 2572.

- SNMP-USER-BASED-SM-MIB

  This MIB module contains management information definitions for the the User Based security model of SNMPv3.
  This MIB module is defined in RFC 2574.
- SNMP-VIEW-BASED-ACM-MIB

  This MIB module contains management information definitions for the View based Access Control Model for SNMPv3.
  This MIB module is defined in RFC 2575.

4.2 Application specific, system specific MIB modules

- WWW-MIB

  This MIB module provides management information for WWW-services. The MIB module is generic and via appropriate mappings multiple protocols can be managed by this MIB module. The protocols possible are those that are based upon a request/response paradigm and deal with document/information transfer.
  This MIB module is defined in RFC 2594.
- NETWORK-SERVICES-MIB

  This MIB module provides management information for networked applications. It provides both administrative management information of the application and network associations (connections) of the application.
  This MIB module is defined in RFC 2788.
- MIB-II

  MIB-2 is the most commonly used MIB module. It provides managed objects for the TCP/IP (UDP) protocol stack, for interfaces available on systems. This MIB module is also mostly used for monitoring connection.
  Nowadays this MIB module evolved in separate MIB modules.
- RMON-MIB

  This MIB module has a framework with which protocols can be monitored going over a network. It functionality is close to 'tcpdump' and/or 'snoop' and its output has an SNMP interface.
- APM-MIB

  The Application Protocol Monitoring MIB module provides performance metrics for application protocols and in particular for HTTP within the Remote MONitoring framework.
- HOST-RESOURCES-MIB

  The MIB module provides management information of the hardware and software of host systems. It can be used to provide for instance CPU load, disk usage and installed applications.
- SYSAPPL-MIB

  The SYSAPPL-MIB module provides management information of application that are installed, running and have been running on a host. This MIB module uses a process oriented view for the running and have been running applications on a system. This management information can be found in a limited form in the HOST-RESOURCES-MIB.
- APPLICATION-MIB

  The APPLICATION-MIB provides more detailed management information of currently running applications on a host system. The MIB module has a process oriented view of the the applications. The information provides for instance the amount of connections and the activity on each of the connections. This MIB module is very useful within the Apache server. However, the burden is high compared to the NETWORK-SERVICES-MONITORING-MIB, since the SYSAPPL-MIB needs to be implemented too.

5.0 The WWW-MIB module

The WWW-MIB maintains management information for WWW services. The WWW service is conceptually defined as a set of actions that can be invoked on information resources. The information resources used in the WWW-MIB are termed documents and the set of actions are specified as an abstract document transfer protocol. The WWW services can be provided by either a client, a server or a proxy. Clients initiates requests to or a server or a proxy. The server would respond to the request of clients and have usually the requested documents on a local information store. The proxy is a special kind of WWW service that acts as server and client. A proxy might translate requests and responses into different protocols and it might also interact with other information retrieval system, for example databases. A special kind of proxy, the caching proxy, is also recognized. No specific managed objects are defined for a caching proxy. The reason for this is that from a protocol perspective the caching proxy is not different, except for the fact that local copies of information resources may be used to responds to requests.

5.1 Structure of the MIB module

A WWW service is represented by three groups of managed objects in the MIB module

- service information
- protocol statistics
- document statistics

5.1.1 Service Information Group

The service information provides the administrative management information for (potentially) multiple WWW services maintained within a single host. WWW services defined as a virtual domain are also maintained. The information is defined within one table, the **wwwServiceTable** and may be divided into two main groups:

- administrative information of the WWW service, such as contact person, and
- network information, such as the protocol used for the WWW service.

Example of the wwwServiceTable

Below is provided an example of the information maintain in this group - the **wwwServiceTable** - and what the MIB module definition is for that information.

The information retrieved from an SNMP agent that has implemented this MIB module is shown below. It shows a general description, system contact, the protocol (here tcp port 80, the well-knwon port for HTTP), the service name, the type of the service, when did the service start, is operational status and when the service status was last changed.

For instance the managed object, **wwwServiceContact** provides contact information for someone who is responsible for this service. One can use therefore, the DESCRIPTION-clause of the defined OBJECT-TYPE of **wwwServiceContact** that is described in the SMI below. The access to this object is always **read-only** (the ACCESS-clause) that means a manager cannot change this. The SNMP agent is responsible for providing the value. The value provided by the SNMP agent is always a '**Utf8String**', since this is defined in the SYNTAX-clause.

One can see that that the OIDs (which is before the '=' sign) of the returned managed objects all are translated into names. This translation is for human readability. However, the last part of the OID - here **.1** - is not translated. This is the index of the table that uniquely identifies all entries in the table. The correct syntax and semantics of the index can be retrieved from the OBJECT-TYPE **wwwServiceEntry** where in INDEX-clause is defined with the value **wwwServiceIndex**. The syntax and semantics can be read in the OBJECT-TYPE of the **wwwServiceIndex** that specifies it is an 32-bit unsinged integer.

SNMP agent output

```
wwwServiceTable.wwwServiceEntry.wwwServiceDescription.1 = Apache/1.3.12 (Unix) SNMP
wwwServiceTable.wwwServiceEntry.wwwServiceContact.1 = webmaster@foo.bar
wwwServiceTable.wwwServiceEntry.wwwServiceProtocol.1 = OID: tcp.80
wwwServiceTable.wwwServiceEntry.wwwServiceName.1 = www.foo.bar
wwwServiceTable.wwwServiceEntry.wwwServiceType.1 = wwwServer(2)
wwwServiceTable.wwwServiceEntry.wwwServiceStartTime.1 = 2001-3-12,17:2:1.0
wwwServiceTable.wwwServiceEntry.wwwServiceOperStatus.1 = running(2)
wwwServiceTable.wwwServiceEntry.wwwServiceLastChange.1 = 2001-3-12,17:2:1.0
```

start of SMI

```
  wwwServiceTable OBJECT-TYPE
      SYNTAX      SEQUENCE OF WwwServiceEntry
      MAX-ACCESS  not-accessible
      STATUS      current
      DESCRIPTION
          "The table of the WWW services known by the SNMP agent."
      ::= { wwwService 1 }

   wwwServiceEntry OBJECT-TYPE
      SYNTAX      WwwServiceEntry
```

```
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "Details about a particular WWW service."
        INDEX       { wwwServiceIndex }
        ::= { wwwServiceTable 1 }

    WwwServiceEntry ::= SEQUENCE {
        wwwServiceIndex             Unsigned32,
        wwwServiceDescription       Utf8String,
        wwwServiceContact          Utf8String,
        wwwServiceProtocol          OBJECT IDENTIFIER,
        wwwServiceName              DisplayString,
        wwwServiceType              INTEGER,
        wwwServiceStartTime         DateAndTime,
        wwwServiceOperStatus        WwwOperStatus,
        wwwServiceLastChange        DateAndTime
    }

    wwwServiceIndex OBJECT-TYPE
        SYNTAX      Unsigned32 (1..4294967295)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "An integer used to uniquely identify a WWW service. The
             value must be the same as the corresponding value of the
             applSrvIndex defined in the Application Management MIB
             (APPLICATION-MIB) if the applSrvIndex object is available.
             It might be necessary to manually configure sub-agents in
             order to meet this requirement."
        ::= { wwwServiceEntry 1 }

    wwwServiceDescription OBJECT-TYPE
        SYNTAX      Utf8String
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "Textual description of the WWW service. This shall include
             at least the vendor and version number of the application
             realizing the WWW service. In a minimal case, this might
             be the Product Token (see RFC 2068) for the application."
        ::= { wwwServiceEntry 2 }

    wwwServiceContact OBJECT-TYPE
        SYNTAX      Utf8String
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The textual identification of the contact person for this
             service, together with information on how to contact this
             person. For instance, this might be a string containing an
             email address, e.g. '<webmaster@domain.name>'."
        ::= { wwwServiceEntry 3 }

    wwwServiceProtocol OBJECT-TYPE
        SYNTAX      OBJECT IDENTIFIER
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "An identification of the primary protocol in use by this
             service. For Internet applications, the IANA maintains
             a registry of the OIDs which correspond to well-known
            application protocols.  If the application protocol is not
            listed in the registry, an OID value of the form
            {applTCPProtoID port} or {applUDPProtoID port} are used for
            TCP-based and UDP-based protocols, respectively. In either
            case 'port' corresponds to the primary port number being
            used by the protocol."
        REFERENCE
           "The OID values applTCPProtoID and applUDPProtoID are
            defined in the NETWORK-SERVICES-MIB (RFC 2248)."
        ::= { wwwServiceEntry 4 }

    wwwServiceName OBJECT-TYPE
        SYNTAX      DisplayString
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "The fully qualified domain name by which this service is
            known. This object must contain the virtual host name if
            the service is realized for a virtual host."
        ::= { wwwServiceEntry 5 }

    wwwServiceType OBJECT-TYPE
        SYNTAX      INTEGER {
                        wwwOther(1),
                        wwwServer(2),
                        wwwClient(3),
                        wwwProxy(4),
                        wwwCachingProxy(5)
                    }
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "The application type using or realizing this WWW service."
        ::= { wwwServiceEntry 6 }

    wwwServiceStartTime OBJECT-TYPE
        SYNTAX      DateAndTime
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "The date and time when this WWW service was last started.
            The value SHALL be '0000000000000000'H if the last start
            time of this WWW service is not known."
        ::= { wwwServiceEntry 7 }

    wwwServiceOperStatus OBJECT-TYPE
        SYNTAX      WwwOperStatus
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "Indicates the operational status of the WWW service."
        ::= { wwwServiceEntry 8 }

    wwwServiceLastChange OBJECT-TYPE
        SYNTAX      DateAndTime
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "The date and time when this WWW service entered its current
            operational state. The value SHALL be '0000000000000000'H if
            the time of the last state change is not known."
        ::= { wwwServiceEntry 9 }
```

end of SMI

5.1.2 Protocol Statistics Group

The protocol statistics provide information regarding the the invoked requests upon the information resources. This group maintains the network management information of the traffic received or transmitted by a WWW service. All the protocol interactions are mapped via the abstracts document transfer protocol. The MIB module distinguishes between outgoing and incoming requests and responses. This makes it possible to obtain statistics for clients, servers and proxies with a single set of managed objects.

The following tables presenting the protocol information are defined:

- The **wwwSummaryTable** provides a summarization of the network traffic and protocol operations related to the associated WWW service. Certain variables are redundant with respect to the request and response tables, but they are added to provide an operator a quick overview and to reduce SNMP network traffic.
- The **wwwRequestInTable** provides detailed information about incoming requests. Every particular request type is counted separately.
- The **wwwRequestOutTable** provides detailed information about outgoing requests. Every particular request type is counted separately.
- The **wwwResponseInTable** provides detailed information about incoming responses. Every particular response type is counted separately.
- The **wwwResponseOutTable** provides detailed information about outgoing responses. Every particular response type is counted separately.

Example of the wwwSummaryTable

Below is provided an example of the information maintain in this group - the '**wwwSummaryTable**' - and what the MIB module definition is for that information.

The information retrieved from an SNMP agent that has implemented this MIB module is shown below. It shows the service summary counters.

For instance the managed object, '**wwwSummaryInRequests**' provides the amount of requests invoked on the service. One can use therefore, the DESCRIPTION-clause of the defined OBJECT-TYPE of **wwwSummaryInRequests** that is described in the SMI below. The access to this object is always **read-only** (the ACCESS-clause) that means a manager cannot change this. The SNMP agent is responsible for providing the value. The value provided by the SNMP agent is always a 32-bit counter, this is defined in the SYNTAX-clause.

One can see that that the OIDs (which is before the '=' sign) of the returned managed objects all are translated into names. This translation is for human readability. However, the last part of the OID - here **.1** - is not translated. This is the index of the table that uniquely identifies all entries in the table. The correct syntax and semantics of the index can be retrieved from the OBJECT-TYPE **wwwSummaryEntry** where in INDEX-clause is defined with the value **wwwServiceIndex**. The syntax and semantics can be read in the OBJECT-TYPE of the **wwwServiceIndex** that specifies it is an 32-bit unsinged integer (NOTE: the wwwServiceIndex was defined in the **wwwServiceTable** shown previously).

SNMP agent output

```
wwwSummaryTable.wwwSummaryEntry.wwwSummaryInRequests.1 = Counter32: 29
wwwSummaryTable.wwwSummaryEntry.wwwSummaryOutRequests.1 = Counter32: 0
wwwSummaryTable.wwwSummaryEntry.wwwSummaryInResponses.1 = Counter32: 0
wwwSummaryTable.wwwSummaryEntry.wwwSummaryOutResponses.1 = Counter32: 29
wwwSummaryTable.wwwSummaryEntry.wwwSummaryInLowBytes.1 = Counter32: 0
wwwSummaryTable.wwwSummaryEntry.wwwSummaryOutLowBytes.1 = Counter32: 43879
```

start of SMI

```
   wwwSummaryTable OBJECT-TYPE
       SYNTAX      SEQUENCE OF WwwSummaryEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "The table providing overview statistics for the
            WWW services on this system."
```

```
        ::= { wwwProtocolStatistics 1 }

    wwwSummaryEntry OBJECT-TYPE
        SYNTAX      WwwSummaryEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "Overview statistics for an individual service."
        INDEX       { wwwServiceIndex }
        ::= { wwwSummaryTable 1 }

    WwwSummaryEntry ::= SEQUENCE {
        wwwSummaryInRequests        Counter32,
        wwwSummaryOutRequests       Counter32,
        wwwSummaryInResponses       Counter32,
        wwwSummaryOutResponses      Counter32,
        wwwSummaryInBytes           Counter64,
        wwwSummaryInLowBytes        Counter32,
        wwwSummaryOutBytes          Counter64,
        wwwSummaryOutLowBytes       Counter32
    }

    wwwSummaryInRequests OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of requests successfully received."
        ::= { wwwSummaryEntry 1 }

    wwwSummaryOutRequests OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of requests generated."
        ::= { wwwSummaryEntry 2 }

    wwwSummaryInResponses OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of responses successfully received."
        ::= { wwwSummaryEntry 3 }

    wwwSummaryOutResponses OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of responses generated."
        ::= { wwwSummaryEntry 4 }

    wwwSummaryInBytes OBJECT-TYPE
        SYNTAX      Counter64
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of content bytes received."
        ::= { wwwSummaryEntry 5 }
    wwwSummaryInLowBytes OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The lowest thirty-two bits of wwwSummaryInBytes."
        ::= { wwwSummaryEntry 6 }

    wwwSummaryOutBytes OBJECT-TYPE
        SYNTAX      Counter64
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of content bytes transmitted."
        ::= { wwwSummaryEntry 7 }

    wwwSummaryOutLowBytes OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The lowest thirty-two bits of wwwSummaryOutBytes."
        ::= { wwwSummaryEntry 8 }
```

end of SMI

5.1.3 Document Statistics Group

The document statistics provide management information on what information resources, documents, have been accessed on the WWW service. This information typically relates to the URL that is access. The information presented is targeted to provide a short term operational management information. Therefore, it must be noted that the document statistics group is not well suited for accounting purposes. This group provides two sorts of views on your document access of a WWW service.

1. A Last N documents view provides a list of N documents that were last accessed for the WWW service. This table provides view that is similar to a 'tail -f' on the access log file.
2. Time interval based document access summaries. An overall summary of the amount of document accessed, amount data bytes is provided as well top N statistics for the documents. A top N statistics is a hit list that is ordered by the amount a particular document was accessed and amount of data bytes that were transferred due to the document access. The time interval based statistics are gathered via a bucket mechanism in which document accesses are stored for the interval upon which at the end of the time interval the statistics are computed.

The following tables are presenting the document statistics:

- The **wwwDocCtrlTable** provides the manager managed objects with which the document statistic tables can be controlled in size, creation and expiration of buckets.
- The **wwwDocLastNTable** provides a view of the last N documents which where accessed. Along with the document some protocol information associated is provided.
- The **wwwDocBucketTable** provides summary statistics information of the time interval the lists the buckets of statistical information that have been collected. An entry in the wwwDocBucketTable contains the creation time stamp of the bucket as well as summary information (number of accesses, number of documents accessed and number of bytes transferred).
- The **wwwDocAccessTopNTable** provides the top N documents that were accessed during the time interval of collection. The order in the table is determined by the amount of accesses for the individual documents.
- The **wwwDocBytesTopNTable** provides the top N documents that were accessed during the time interval of collection. The order in the table is determined by the amount of bytes transferred for the individual documents.

Example of the wwwDocBucketTable

Below is provided an example of the information maintain in this group - the **wwwDocBucketTable** - and what the MIB module definition is for that information.

The information retrieved from an SNMP agent that has implemented this MIB module is shown below. It shows the service summary counters.

For instance the managed object, **wwwDocBucketTimeStamp** provides the date and time when this bucket was created. One can use therefore, the DESCRIPTION-clause of the defined OBJECT-TYPE of **wwwDocBucketTimeStamp** that is described in the SMI below. The access to this object is always **read-only** (the ACCESS-clause) that means a manager cannot change this. The SNMP agent is responsible for providing the value. The value provided by the SNMP agent is always a 32-bit counter, this is defined in the SYNTAX-clause.

One can see that that the OIDs (which is before the '=' sign) of the returned managed objects all are translated into names. This translation is for human readability. However, the last part of the OID - here **.1.1** - is not translated. This is the index of the table that uniquely identifies all entries in the table. The correct syntax and semantics of the index can be retrieved from the OBJECT-TYPE **wwwDocBucketEntry** where in INDEX-clause is defined with the value **wwwServiceIndex** and the **wwwDocBucketIndex**. The syntax and semantics can be read in the OBJECT-TYPE of respecitvely the **wwwServiceIndex** and the **wwwDocBucketIndex**. The **wwwServiceIndex** was defined in the **wwwServiceTable** shown previously. The **wwwDocBucketIndex** is defined below with the SYNTAX-clause **Unsigned32** that is a 32-bit unsigned integer.

SNMP agent output

```
wwwDocBucketTable.wwwDocBucketEntry.wwwDocBucketTimeStamp.1.1 = 2001-3-12,18:25:38.0
wwwDocBucketTable.wwwDocBucketEntry.wwwDocBucketAccesses.1.1 = Gauge32: 47
wwwDocBucketTable.wwwDocBucketEntry.wwwDocBucketDocuments.1.1 = Gauge32: 24
wwwDocBucketTable.wwwDocBucketEntry.wwwDocBucketBytes.1.1 = Gauge32: 71515
```

start of SMI

```
wwwDocBucketTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF WwwDocBucketEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table provides administrative summary information for
         the buckets maintained per WWW service."
    ::= { wwwDocumentStatistics 3 }

wwwDocBucketEntry OBJECT-TYPE
    SYNTAX      WwwDocBucketEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An entry which describes the parameters associated with a
         particular bucket."
    INDEX       { wwwServiceIndex, wwwDocBucketIndex }
    ::= { wwwDocBucketTable 1 }

WwwDocBucketEntry ::= SEQUENCE {
    wwwDocBucketIndex           Unsigned32,
    wwwDocBucketTimeStamp       DateAndTime,
    wwwDocBucketAccesses        Unsigned32,
    wwwDocBucketDocuments       Unsigned32,
    wwwDocBucketBytes           Unsigned32
}

wwwDocBucketIndex OBJECT-TYPE
    SYNTAX      Unsigned32 (1..4294967295)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An arbitrary monotonically increasing integer number
         used for indexing the wwwDocBucketTable. The index number
         wraps to 1 whenever the maximum value is reached."
    ::= { wwwDocBucketEntry 1 }

wwwDocBucketTimeStamp OBJECT-TYPE
```

```
        SYNTAX      DateAndTime
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The date and time when the bucket was made available."
        ::= { wwwDocBucketEntry 2 }

    wwwDocBucketAccesses OBJECT-TYPE
        SYNTAX      Unsigned32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The total number of access attempts for any document
             provided by this WWW service during the time interval
             over which this bucket was created."
        ::= { wwwDocBucketEntry 3 }

    wwwDocBucketDocuments OBJECT-TYPE
        SYNTAX      Unsigned32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The total number of different documents for which access
             was attempted this this WWW service during the time interval
             over which this bucket was created."
        ::= { wwwDocBucketEntry 4 }

    wwwDocBucketBytes OBJECT-TYPE
        SYNTAX      Unsigned32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The total number of content bytes which were transferred
             from this WWW service during the time interval over which
             this bucket was created."
        ::= { wwwDocBucketEntry 5 }
```

end of SMI

6.0 The NETWORK-SERVICES-MIB module.

The NETWORK-SERVICES-MIB module provide a generic set of managed objects for application level management. The network management information provided by this MIB module is divided into three parts.

1. Application administrative information that provides descriptive information of the application. For instance, the application name.
2. Application layer specific network information that provides, for instance, the total amount of connections the application had and the current amount of network connections active.
3. Connection specific information providing detailed information of connections the application currently maintains.

The following tables are defined in the MIB module:

- The 'applTable' provides administrative information and summarizes network activity of the application.
- The 'assocTable' provides connection specific information that the application currently has active/open.

Example of the wwwDocBucketTable

Below is provided an example of the information maintain in this group - the **wwwDocBucketTable** - and what the MIB module definition is for that information.

The information retrieved from an SNMP agent that has implemented this MIB module is shown below. It shows the service summary counters.

For instance the managed object, **applInboundAssociations** provides the number of current connections to this application. For the syntax and semantics is referred to the DESCRIPTION-clause of the defined OBJECT-TYPE of **applInboundAssociations** that is described in the SMI below. The access to this object is always **read-only** (the ACCESS-clause) that means a manager cannot change this. The SNMP agent is responsible for providing the value. The value provided by the SNMP agent is always a 32-bit gauge, this is defined in the SYNTAX-clause.

One can see that that the OIDs (which is before the '=' sign) of the returned managed objects all are translated into names. This translation is for human readability. However, the last part of the OID - here **.546** - is not translated. This is the index of the table that uniquely identifies all entries in the table. The correct syntax and semantics of the index can be retrieved from the OBJECT-TYPE **applEntry** where in INDEX-clause is defined with the value **applIndex**. The syntax and semantics can be read in the OBJECT-TYPE of the **applIndex**. The **applIndex** is defined below with the SYNTAX-clause **Integer** with range that is equal to a 32-bit unsigned integer.

SNMP agent output

```
applTable.applEntry.applName.546 = Apache HTTP server
applTable.applEntry.applDirectoryName.546 = ""
applTable.applEntry.applVersion.546 = Apache/1.3.12 (Unix) ConductorSNMP/1.0.3
applTable.applEntry.applUptime.546 = Timeticks: (0) 0:00:00.00
applTable.applEntry.applOperStatus.546 = up(1)
applTable.applEntry.applLastChange.546 = Timeticks: (0) 0:00:00.00
applTable.applEntry.applInboundAssociations.546 = Gauge32: 13
applTable.applEntry.applOutboundAssociations.546 = Gauge32: 0
applTable.applEntry.applAccumulatedInboundAssociations.546 = Counter32: 658
applTable.applEntry.applAccumulatedOutboundAssociations.546 = Counter32: 0
applTable.applEntry.applLastOutboundActivity.546 = Timeticks: (0) 0:00:00.00
applTable.applEntry.applRejectedInboundAssociations.546 = Counter32: 0
applTable.applEntry.applFailedOutboundAssociations.546 = Counter32: 0
applTable.applEntry.applDescription.546 = World Wide Web application (server-side)
applTable.applEntry.applURL.546 = http://www.apache.org/
```

start of SMI

```
applTable OBJECT-TYPE
    SYNTAX SEQUENCE OF ApplEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "The table holding objects which apply to all different
         kinds of applications providing network services.
         Each network service application capable of being
         monitored should have a single entry in this table."
    ::= {application 1}

applEntry OBJECT-TYPE
    SYNTAX ApplEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
      "An entry associated with a single network service
       application."
    INDEX {applIndex}
    ::= {applTable 1}

ApplEntry ::= SEQUENCE {
    applIndex                              INTEGER,
    applName                               SnmpAdminString,
    applDirectoryName                      DistinguishedName,
    applVersion                            SnmpAdminString,
    applUptime                             TimeStamp,
    applOperStatus                         INTEGER,
    applLastChange                         TimeStamp,
    applInboundAssociations                Gauge32,
    applOutboundAssociations               Gauge32,
    applAccumulatedInboundAssociations     Counter32,
    applAccumulatedOutboundAssociations    Counter32,
    applLastInboundActivity                TimeStamp,
    applLastOutboundActivity               TimeStamp,
    applRejectedInboundAssociations        Counter32,
    applFailedOutboundAssociations         Counter32,
    applDescription                        SnmpAdminString,
    applURL                                URLString
}

applIndex OBJECT-TYPE
    SYNTAX INTEGER (1..2147483647)
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
      "An index to uniquely identify the network service
       application. This attribute is the index used for
       lexicographic ordering of the table."
    ::= {applEntry 1}

applName OBJECT-TYPE
    SYNTAX SnmpAdminString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      "The name the network service application chooses to be
       known by."

    ::= {applEntry 2}

applDirectoryName OBJECT-TYPE
    SYNTAX DistinguishedName
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      "The Distinguished Name of the directory entry where
       static information about this application is stored.
       An empty string indicates that no information about
       the application is available in the directory."
    ::= {applEntry 3}

applVersion OBJECT-TYPE
    SYNTAX SnmpAdminString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      "The version of network service application software.
       This field is usually defined by the vendor of the
       network service application software."
    ::= {applEntry 4}
applUptime OBJECT-TYPE
    SYNTAX TimeStamp
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      "The value of sysUpTime at the time the network service
       application was last initialized.  If the application was
       last initialized prior to the last initialization of the
       network management subsystem, then this object contains
       a zero value."
    ::= {applEntry 5}

applOperStatus OBJECT-TYPE
    SYNTAX INTEGER {
      up(1),
      down(2),
      halted(3),
      congested(4),
      restarting(5),
      quiescing(6)
    }
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      "Indicates the operational status of the network service
       application. 'down' indicates that the network service is

       not available. 'up' indicates that the network service
       is operational and available.  'halted' indicates that the
       service is operational but not available.  'congested'
       indicates that the service is operational but no additional
       inbound associations can be accommodated.  'restarting'
       indicates that the service is currently unavailable but is
       in the process of restarting and will be available soon.
       'quiescing' indicates that service is currently operational
       but is in the process of shutting down. Additional inbound
       associations may be rejected by applications in the
       'quiescing' state."
    ::= {applEntry 6}

applLastChange OBJECT-TYPE
    SYNTAX TimeStamp
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      "The value of sysUpTime at the time the network service
       application entered its current operational state.  If
       the current state was entered prior to the last
       initialization of the local network management subsystem,
       then this object contains a zero value."
    ::= {applEntry 7}

applInboundAssociations OBJECT-TYPE
    SYNTAX Gauge32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      "The number of current associations to the network service
       application, where it is the responder.  An inbound
       association occurs when another application successfully
       connects to this one."
    ::= {applEntry 8}

applOutboundAssociations OBJECT-TYPE
    SYNTAX Gauge32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      "The number of current associations to the network service
       application, where it is the initiator.  An outbound
       association occurs when this application successfully
       connects to another one."
    ::= {applEntry 9}

applAccumulatedInboundAssociations OBJECT-TYPE

    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      "The total number of associations to the application entity
       since application initialization, where it was the responder."
    ::= {applEntry 10}

applAccumulatedOutboundAssociations OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      "The total number of associations to the application entity
       since application initialization, where it was the initiator."
    ::= {applEntry 11}

applLastInboundActivity OBJECT-TYPE
    SYNTAX TimeStamp
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      "The value of sysUpTime at the time this application last
       had an inbound association.  If the last association
       occurred prior to the last initialization of the network
       subsystem, then this object contains a zero value."
    ::= {applEntry 12}

applLastOutboundActivity OBJECT-TYPE
    SYNTAX TimeStamp
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      "The value of sysUpTime at the time this application last
       had an outbound association.  If the last association
       occurred prior to the last initialization of the network
       subsystem, then this object contains a zero value."
    ::= {applEntry 13}

applRejectedInboundAssociations OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      "The total number of inbound associations the application
       entity has rejected, since application initialization.
       Rejected associations are not counted in the accumulated
       association totals.  Note that this only counts

       associations the application entity has rejected itself;
       it does not count rejections that occur at lower layers
       of the network.  Thus, this counter may not reflect the
       true number of failed inbound associations."
    ::= {applEntry 14}

applFailedOutboundAssociations OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      "The total number associations where the application entity
       is initiator and association establishment has failed,
       since application initialization.  Failed associations are
       not counted in the accumulated association totals."
    ::= {applEntry 15}

applDescription OBJECT-TYPE
    SYNTAX SnmpAdminString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      "A text description of the application.  This information
       is intended to identify and briefly describe the
       application in a status display."
    ::= {applEntry 16}

applURL OBJECT-TYPE
    SYNTAX URLString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
      "A URL pointing to a description of the application.
       This information is intended to identify and describe
       the application in a status display."
    ::= {applEntry 17}
```

end of SMI

7.0 Conclusion

The Internet Management Framework also known as SNMP can assist in managing your Apache HTTP server. The framework consists of managers and agents. The agents contain a MIB that presents the managed instrumentation as managed objects. SNMP is an open standard and that reduces the cost of ownership. Many companies support the standard and implement it in many products and management applications.

The component that represent the managed instrumentation is the MIB. It consists of a collection of managed objects that are defined in SMI. With SMI one can either define scalar objects or columnar objects. Scalar objects or used for values that system wide only exist once. The columnar object are ordered in a conceptual table. The action upon the MIB can be either on the exact object or on the next object.

Already many MIB modules are defined within standardization organizations. These MIB modules supported already various components of networked systems. Two MIB modules are applicable to the Apache HTTP server. The WWW-MIB provides managed objects with a WWW service oriented view. The NETWORK-SERVICES-MIB module provides managed objects for the Apache as an Application.

8.0 Contact Information

Author	Harrie Hazewinkel
Postal	Covalent Technologies, Inc. 706 Mission Street (second floor) San Francisco, CA - 94103 USA
Email	harrie@covalent.net

Contributed to the Apache Software Foundation by Harrie Hazewinkel, Covalent Technologies, Inc..

Appendix A: The SNMP protocol operations

This appendix provides the basic SNMP protocol operations. It serves the purpose on how management information is conveyed between managers and agents. The SNMP protocol operations invoke actions upon the MIB residing in the SNMP agent. Therefore, the following figure depicts the MIB tree. The MIB tree is used for the example commands made for the protocol actions.

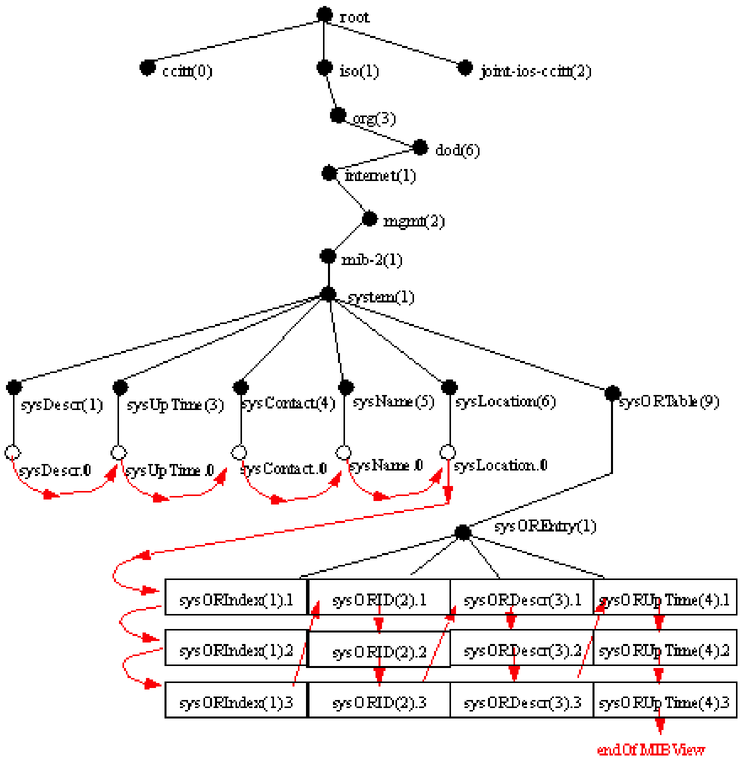

Figure A-1: example MIB

SNMP GET operation

The SNMP-GET operation is invoked by the manager to retrieve an exact list of variables which must exist in the MIB of the SNMP agent. If a management application requests a variable that does not exist in the MIB the SNMP agent will return a 'noSuchName' for that variable including the error status set. The following figure A-2 depicts the SNMP operation, where the manager invokes an SNMP-GET and the agent responds to it with an SNMP-RESPONSE.
The following parameters are used in the SNMP-PDU.

SNMP-GET:
PDU type: GetRequest-PDU
Request-ID: Sequence number
Variable bindings: A list of variables (each variable consists of {identity, nullType, nullValue})

SNMP-RESPONSE:
PDU type: Response-PDU
Request-ID: Sequence number
Error Status: The type of error returned for the variable pointed by 'error-index'
Error Index: The index of the variable at fault in the variable binding list.
Variable bindings: A list of variables (each variable consists of {identity, type, value})

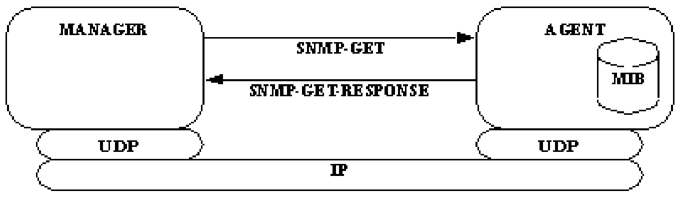

Figure A-2: SNMP-GET operation

Examples for the SNMP-GET

In the first request example we see that we get indeed the value required. The second example shows us that we did not asked for the leaf of the MIB tree. Therefore, the error-status and error-index is set and the values of the requested objects failed. The request asked a value of a node in the MIB tree that was not an instance and contained a value.

Request	Response
varbind: • sysDescr.0 • sysUpTime.0	error-status: 0 error-index: 0 varbind: • system.sysDescr.0 = "FreeBSD host 4.1- RELEASE" • system.sysUpTime.0 = Timeticks: (420250) 1:10:02.50
varbind: • sysDescr • sysUpTime	error-status: noSuchName error-index: 1 varbind: • system.sysDescr = retrieval failed • system.sysUpTime = retrieval failed

SNMP GETNEXT operation

The SNMP-GET operation is invoked by the manager to retrieve a list of variables that are the lexical graphical next variables in the MIB of the SNMP agent. If a management application requests a variable after the last variable in the MIB the SNMP agent will return a 'endOfMIB' for that variable including the error status set. The following figure A-3 depicts the SNMP operation, where the manager invokes an SNMP-GETNEXT and the agent responds to it with an SNMP-RESPONSE. This operations is very useful if the management application has knowledge of the approximation of where the needed variable in the MIB resides.
The following parameters are used in the SNMP-PDU.

SNMP-GETNEXT:
PDU type: GetNextRequest-PDU
Request-ID: Sequence number
Variable bindings: A list of variables (each variable consists of {identity, nullType, nullValue})

SNMP-RESPONSE:
PDU type: Response-PDU
Request-ID: Sequence number
Error Status: The type of error returned for the variable pointed by 'error-index'
Error Index: The index of the variable at fault in the variable binding list.
Variable bindings: A list of variables which are the lexical graphical next of the variables requested. next to the each variable consists of {identity, type, value})

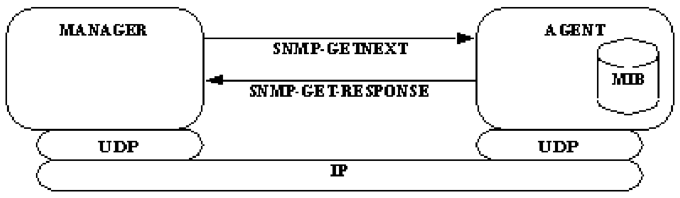

Figure A-3: SNMP-GETNEXT operation

Examples for the SNMP-GETNEXT

In the first request example we see that we do not get the **sysDescr.0** and the **sysUptime.0** objects. We get indeed the lexical graphical next in the MIB tree. The second example shows us that we did not asked for the leaf of the MIB tree, but still get the **sysDescr.0** and the **sysUptime.0** objects because we requested an node previous in the tree.

Request	Response
varbind: • sysDescr.0 • sysUpTime.0	error-status: 0 error-index: 0 varbind: • system.sysUpTime.0 = Timeticks: (420250) 1:10:02.50 • system.sysContact.0 = "<webmaster@your.domain>"
varbind: • sysDescr • sysUpTime	error-status: 0 error-index: 0 varbind: • system.sysDescr.0 = "FreeBSD host 4.1- RELEASE" • system.sysUpTime.0 = Timeticks: (420250) 1:10:02.50

SNMP GETBULK operation

The SNMP-GETBULK operation is invoked by the manager to retrieve an exact list of variables with which you can retrieve multiple variables by asking for 1. Additional parameters need to be provided with the request and then 1 or more next available variable bindings can be retrieved. The following figure A-4 depicts the SNMP operation, where the manager invokes an SNMP-GETBULK and the agent responds to it with an SNMP-RESPONSE.

NOTE: This operation is not supported in SNMPv1.

The following parameters are used in the SNMP-PDU.

SNMP-GETBULK:
PDU type: GetBulkRequest-PDU
Request-ID: Sequence number
Non-repeaters: The amount of variables bindings starting from the beginning of the list are requested only the next object in the MIB.
Max-repetitions: The maximum lexigraphical next objects are are requested for the remainder of the variable bindings.
Variable bindings: A list of variables (each variable consists of {identity, nullType, nullValue})

SNMP-RESPONSE:
PDU type: Response-PDU
Request-ID: Sequence number
Error Status: The type of error returned for the variable pointed by 'error-index'
Error Index: The index of the variable at fault in the variable binding list.
Variable bindings: A list of variables (each variable consists of {identity, type, value})

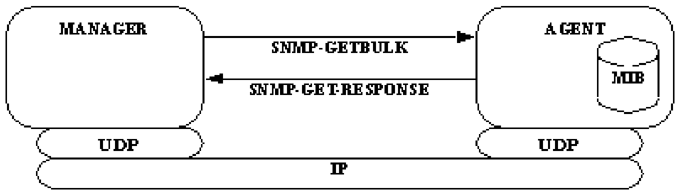

Figure A-4: SNMP-GETBULK operation

Examples for the SNMP-GETBULK

In the example we see that the we request 1 non-repeater which will be the next available object for the first varbind and 3 repetitions maximum for the rest of the varbinds - here sysORIndex. The response we see that we retrieve we do not get the **sysDescr.0** and the **sysUptime.0** objects. We get indeed the lexical graphical next in the MIB tree. The second example shows us that we did not asked for the leaf of the MIB tree, but still get the **sysDescr.0** and the **sysUptime.0** objects because we requested an node previous in the tree.

Request	Response
non-repeaters: 1 max repetitions: 3 varbind: • sysUpTime • sysORIndex	error-status: 0 error-index: 0 varbind: • sysUpTime.0 = Timeticks: (759285) 2:06:32.85 • sysORID.1 = OID: snmpMIB • sysORID.2 = OID: snmpVacmMIB.vacmMIBConformance.vacmMIBGroups.vacmBasicGroup • sysORID.3 = OID: snmpFrameworkMIB.snmpFrameworkMIBConformance.snmpFrameworkMIBCompliances.snmpFrameworkMIBComplianc

SNMP SET operation

The SNMP-SET operation is invoked by the manager to change objects within the MIB of the SNMP agent. The following A-5 depicts the SNMP operation, where the manager invokes an SNMP-GET and the agent responds to it with an SNMP-RESPONSE.
The following parameters are used in the SNMP-PDU.

SNMP-SET:
PDU type: SetRequest-PDU
Request-ID: Sequence number
Variable bindings: A list of variables (each variable consists of {identity, type, value})

SNMP-RESPONSE:
PDU type: Response-PDU
Request-ID: Sequence number
Error Status: The type of error returned for the variable pointed by 'error-index'
Error Index: The index of the variable at fault in the variable binding list.
Variable bindings: A list of variables (each variable consists of {identity, type, value})

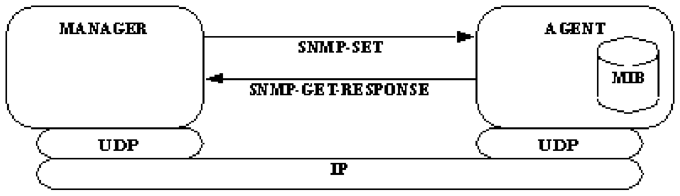

Figure A-5: SNMP-SET operation

Examples for the SNMP-SET

In the first request example we see that we cannot change the value of **sysUpTime.0**. This is because the MIB module definition defines the managed object as read-only. The second example shows that an SNMP-SET is possible. That is because the managed object is defined in the MIB module definition as read-write and the SNMP agent has imlemented it as reqad-write.

Request	Response
varbind: • sysUpTime.0	error-status: not-writable (in SNMP2c) error-index: 1 varbind: • system.sysUpTime.0 = snmpset failed
varbind: • sysContact.0, DisplayString, "new-contact@your.domain"	error-status: 0 error-index: 0 varbind: • system.sysContact.0 = "new-contact@your.domain"

NOTE: Even if a MIB module definition specifies a managed object is read-write, it could well be that the SNMP agent has implemented it as read-only. This is possible when the SNMP agent is not impleemnted in accordance with the conformance statement.

SNMP TRAP operation

The SNMP-TRAP operation is invoked by the SNMP agent to notify the manager of an anomaly that has happened or special events. retrieve an exact list of variables which must exist in the MIB of the SNMP agent. If a management application requests a variable that does not exist in the MIB the SNMP agent will return a 'noSuchName' for that variable including the error status set. The following figure A-6 depicts the SNMP operation, where the manager invokes an SNMP-GET and the agent responds to it with an SNMP-RESPONSE.
The following parameters are used in the SNMP-PDU for SNMPv1.

SNMP-TRAP:
PDU type: SNMP-Trap-PDU
Request-ID: Sequence number
Variable bindings: A list of variables

- first varbind: { enterprise, OBJECT IDENTIFIER, value}
- second varbind: { agent-addres, OCTET STRING, "x.x.x.x"}
- third varbind: {generic trap, INTEGER, value}
- fourth varbind: { specific trap, INTEGER, value}
- fifth varbind: { time stamp, INTEGER, value}
- additional objects that must be defined in the TRAP-TYPE within the MIB module definition {identity, type, value}

The following parameters are used in the SNMP-PDU for SNMPv2.

SNMP-TRAP:
PDU type: SNMPv2-Trap-PDU
Request-ID: Sequence number
Variable bindings: A list of variables

- first varbind: {sysUpTime.0, TimeTicks, value}
- second varbind: { snmpTrapOID.0, OBJECT IDENTIFIER, value}
- additional objects that must be defined in the NOTIFICATION-TYPE within the MIB module definition {identity, type, value}

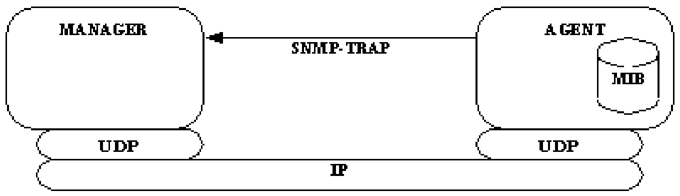

Figure A-6: SNMP-TRAP operation

SNMP INFORM operation

The SNMP-INFORM operation is invoked by the SNMP agent to notify the manager of an anomaly that has happened or special events. retrieve an exact list of variables which must exist in the MIB of the SNMP agent. If a management application requests a variable that does not exist in the MIB the SNMP agent will return a 'noSuchName' for that variable including the error status set. The following figure A-7 depicts the SNMP operation, where the manager invokes an SNMP-GET and the agent responds to it with an SNMP-RESPONSE.
The following parameters are used in the SNMP-PDU for SNMPv2.

SNMP-INFORM:
PDU type: SNMPv2-Inform-PDU
Request-ID: Sequence number
Variable bindings: A list of variables

- first varbind: {sysUpTime.0, TimeTicks, value}
- second varbind: { snmpTrapOID.0, OBJECT IDENTIFIER, value}
- additional objects that must be defined in the NOTIFICATION-TYPE within the MIB module definition {identity, type, value}

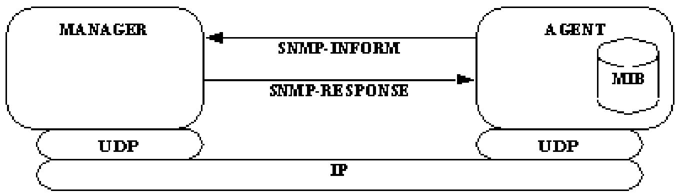

Figure A-7: SNMP-INFORM operation

Appendix B: Object Identifiers of Management information

The managed objects are the variables or instances conveying the management information. The managed objects are ordered in the MIB tree based on the lexical graphical order of the Object Identifier. An instance is determined by the OID prefix of the class and the identification within that class. Two types of instances are identified which are explained in the next 2 section. However, the concept of the OID value identifying a particular instance is equal. Figure B-1 depicts the schematic creation of the OID value.

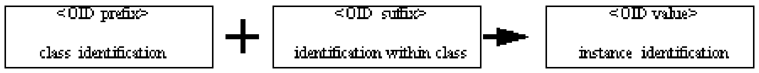

figure B-1: Identification of an SNMP variable

The scalar object

A scalar object is a single instance within a class. These are values that represent the overall system of managed device. Figure B-2 depicts the construction of the OID for a scalar object.

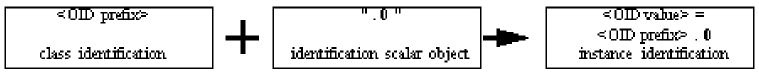

figure B-2 Identification of an scalar variable

Within a MIB module the scalar object is defined by 1 OBJECT-TYPE definition. Below is an example of the system description object provided that is a scalar object.

——— start of SMI ———————————————

```
sysDescr OBJECT-TYPE
    SYNTAX     DisplayString
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
        "A textual description of the entity.  This value should
         include the full name and version identification of the
         system's hardware type, software operating-system, and
         networking software."
        ::= { system  1 }
```

end of SMI

The columnar object

A columnar object is an object that is logically ordered within a table. Within a class there are multiple instances each identified by a separate INDEX used within the OID suffix. The figure 4 on page 3 depicts the construction of a columnar object.

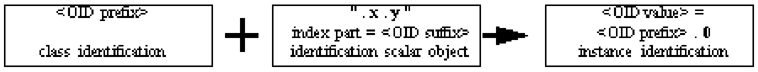

figure 4: Identification of an columnar variable

A columnar object is defined within a logical table. A complete table is defined with the OBJECT-TYPEs and a SEQUENCE definition. The first OBJECT-TYPE defines a table object and has a SYNTAX-clause of 'SEQUENCE OF <name>' which is followed by another OBJECT-TYPE defining the entry object. The entry object has a SYNTAX-clause of '<name>' which is a type defined by a SEQUENCE type assignment. Each columnar object must be specified sportily in an OBJECT-TYPE. Below is an example provided of the

start of SMI

```
    sysORTable OBJECT-TYPE
        SYNTAX     SEQUENCE OF SysOREntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
            "The (conceptual) table listing the capabilities of the
             local SNMPv2 entity acting in an agent role with respect to
             various MIB modules.  SNMPv2 entities having dynamically-
             configurable support of MIB modules will have a
             dynamically-varying number of conceptual rows."
        ::= { system  9 }

    sysOREntry OBJECT-TYPE
        SYNTAX     SysOREntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
            "An entry (conceptual row) in the sysORTable."
        INDEX      { sysORIndex }
        ::= { sysORTable  1 }

    SysOREntry ::= SEQUENCE {
            sysORIndex            INTEGER,
            sysORID               OBJECT IDENTIFIER,
            sysORDescr            DisplayString,
            sysORUpTime           TimeStamp
        }

    sysORIndex OBJECT-TYPE
        SYNTAX     INTEGER
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
            "The auxiliary variable used for identifying instances of
             the columnar objects in the sysORTable."
        ::= { sysOREntry  1 }

    sysORID OBJECT-TYPE
        SYNTAX     OBJECT IDENTIFIER
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
            "An authoritative identification of a capabilities statement
             with respect to various MIB modules supported by the local
             SNMPv2 entity acting in an agent role."
        ::= { sysOREntry  2 }

    sysORDescr OBJECT-TYPE
        SYNTAX     DisplayString
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
            "A textual description of the capabilities identified by the
             corresponding instance of sysORID."
        ::= { sysOREntry  3 }

    sysORUpTime OBJECT-TYPE
        SYNTAX     TimeStamp
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
            "The value of sysUpTime at the time this conceptual row was
             last instanciated."
        ::= { sysOREntry  4 }
```

end of SMI

Naming conventions

The SMI defines a naming scheme to be used for each numbered node. These names are normally human readable names in which most management applications will translate the numbered representation of an object in the MIB tree. However, it is possible that your management application is not aware of the number ot name translation and vice versa. This means in most cases that the managed objects are only presented in an numbered form In order to know which synatx and semantics a managed object an OID in numbered form has one must traverse the MIB tree to do the translation manually by finding the apporaiate name with the OID in numbered form.
The following three OIDs are the well-known named and numbered form from which one could search.

mib-2	1.3.6.1.2.1.
enterprises	1.3.6.1.4.1
snmpModules	1.3.6.1.6.3

Appendix C: Datatypes for network management information

A MIB module has a defined set of datatypes in which network management can be expressed. Various datatypes are defined for the SMI and each type has special semantics and are used for special purposes. The table below specifies all datatypes available and provides an example usage for the datatype.

INTEGER and Integer32	The 'INTEGER' datatype is a integer of undefined maximum and minimum value. However, this type is mainly used only for enumerations. An enumeration is a value naming for specific values. Enumerations are advised to start at 1 and continuously. Each value will have a human readable label.
Integer32	The 'Integer32' datatype is used ot define 32-bit integer values and includes both negative and non negative numbers. The datatype is used in cases where a random number (32-bits) is provided as network management information.
OCTET STRING	The 'OCTET STRING' datatype is used to specify octets (8-bits bytes) of binary or textual information. This datatype is mostly used to provide textual information such as 'sysDescr' which specifies a human readable string for the systems description.
OBJECT IDENTIFIER	The 'OBJECT IDENTIFIER' datatype is used to specify This datatype is mostly used to point to other managed objects or sometimes for specification of types. The types are then defined in a way that other MIB modules can specify type in the form of an OBJECT IDENTIFIER.
IpAddress	The 'IpAddress' is a datatype used to specify the numerical value of an IPv4 address. This value would be used in managed objects that provide the address of an IPv4 host or interface.
Counter32	The 'Counter32' datatype is used to specify an unsigned 32-bit value that includes only non negative numbers. The behavior of this datatype is that the value may only increase. The usage of this datatype is for instance to count all HTTP-requests that has been done. The real value of a counter is the difference over an interval. The real value is never used in network management, only the differences over a time interval.
Unsigned32	The 'Unsigned32' datatype is to provide non negative numbers. This datatype is preferred above the 'Integer32' if it only has non negative numbers.
Gauge32	The 'Gauge32' datatype is used for an non negative number that may increase and decrease. This datatype is used for instance for managed objects that provide 'load information' or the amount of current connections to an application.
TimeTicks	The 'TimeTicks' datatype is used to provide a timer value with the resolution of 1 millisecond. The value is mostly used for time since a certain event or used for a snapshot of the value of the 'sysUpTime'. The 'sysUpTime' is the time since the SNMP entity has started.
Opaque	The 'Opaque' datatype is similar to the OCTET STRING. It is used to convey octets (8-bits bytes) of binary data. However, the use of this datatype is strongly discouraged.
Counter64	The 'Counter64' datatype is used to specify an unsigned 64-bit value that includes only nonnative numbers. The behavior of this datatype is that the value may only increase. The usage of this datatype is required when the 'Counter32' wraps around to often for management application to detect. For instance, the byte count of high speed interfaces would make use of this datatype.
BITS	The 'BITS' datatype is a pseudo type for the OCTET STRING and is used for specifying particular bits within a value. The bits should be numbered continuously and starting at 1. Each definition of a bit will be labeled by a name. This datatype is mostly used when combinations of values (options) are possible. For instance, a managed object that represents the used module hooks of the Apache API. Since there are to many API hooks it is to many provide all combinations of the API hooks. Each API hooks is now represented by a bit label.