RESULTS OF THE CEO PROJECT
WWW MANAGEMENT

HARRIE HAZEWINKEL
ERIC VAN HENGSTUM
AIKO PRAS

CTIT Technical Report Series
No. 96-18
ISSN 1381-3625
November 1996

This document is the result of a proof of concept study that was carried out by ESYS Ltd. and the CTIT of the University of Twente (UT) as part of a programme called the ‘Centre for Earth Observation’ (CEO), which is coordinated by the Joint Research Centre (JRC) of the EC.

Abstract

This report contains the result of a ‘proof of concept’ study that was performed by the CTIT of the University of Twente, together with ESYS Limited (Guildford, UK) for the Institute of Remote Sensing Applications of the Joint Research Centre (JRC) of the EC (Ispra, Italy). The study is part of the ‘Centre of Earth Observation’ (CEO) programme.

The subject of the study was the design and implementation of tools that allow status and utilisation monitoring of networks and distributed information servers. In the specific case of the CEO programme, these information servers are accessible via the WWW and contain large amounts of earth observation data (e.g. satellite pictures).

The work division within the project was that ESYS investigated the management applications, which had to run on top of HP-Openview, and the CTIT designed and implemented the management agents. These agents had to include the following Management Information Bases (MIBs):

- A HTTP-MIB, with detailed information concerning the WWW document transfer protocol.
- A Retrieval Service (RS) MIB, with high level information concerning the WWW document transfer service.
- An Information Store (IS) MIB, with information concerning the WWW server and the documents provided by that server.

The specifications of these MIBs were presented to the IETF and provided a good starting point for subsequent standardization activities. The agents were implemented as sub-agents of the EMANATE extensible agent package and are currently being tested in a number of field trials.

Acknowledgements

This document could not have been written without the help of Mark Gamble, Carl W. Kalbfleisch, Rui Meneses, Jürgen Schönwälder and the other members of the HTTP mailinglist.

Part 3: Definition of the WWW MIBs

Part 5: Conclusions and Recommendations

Part I

Introduction to the World Wide Web

1 Introduction

Due to the rising demand for information by Internet users, information retrieval applications on the Internet are becoming increasingly popular and important. The explosive growth of Internet users and the increased availability of multi-media information pushes the evolution of information retrieval applications, like for instance the World-Wide Web (WWW), to great heights.

This rapid evolution, the large amount of (multi media) data and the diverse nature of the Internet user requirements gives rise to a quite complicated network management problem. To manage the heterogeneous environment, adequate network management instrumentation is required. Unfortunately the existing management instrumentation, which is based on the Simple Network Management Protocol (SNMP), does not yet include the support for managing the aforementioned information retrieval applications.

The Centre of Earth Observations (CEO) is a programme of the European Commission (EC) to promote the use of earth observations data and foster related activities in Europe. The CEO programme is coordinated by the Institute for Remote Sensing Applications (IRSA) of the EC's Joint Research Centre (JRC) at Ispra, Italy.

To support the CEO programme, a set of information retrieval services will be installed that help Internet users to locate and retrieve earth observation data through the use of facilities such as electronic catalogues, subscription services and gateways to existing data bases. The CEO information retrieval services will be distributed across Europe, and will be based upon the existing European Internet infrastructure.
To manage these services, monitoring facilities are needed that help the participants of the CEO programme to improve the (quality of) delivered services and assist information providers and service administrators in finding possible solutions to match quality of service levels to user expectations.

Considering that the WWW has become the favoured mechanism for making information available over the Internet, a method for monitoring the utilisation and performance of WWW Information Servers is a desirable asset for the CEO and one of the features that has been identified for a possible implementation in the CEO enabling Services monitoring facility.

In mid-1995, the CEO has selected ESYS Ltd. (UK) and the Centre for Telematics and Information Technology (CTIT) of the University of Twente (the Netherlands) for the design and implementation of WWW monitoring prototypes. ESYS Ltd. became responsible for the development of a WWW manager application, which should be based on HP OpenView. The University of Twente became responsible for the development of WWW MIBs and agents. The emphasis of this document is on this latter development.

2 WWW technology

The World Wide Web (WWW) is a distributed application with users all over the Internet. It is an information providing / retrieving application which operates on a request/response paradigm, the traditional client/server model. The Internet systems that provide information are called servers; the systems that request information are called clients.

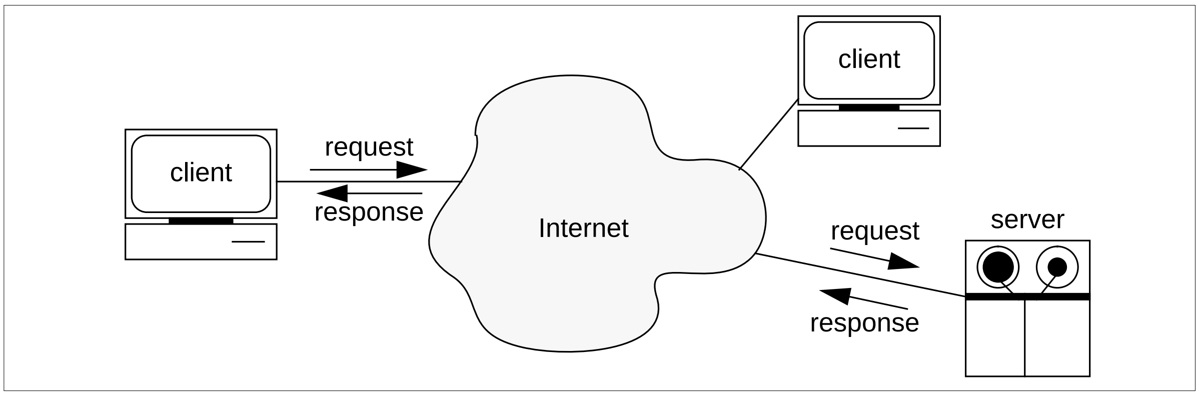


Figure 1: The WWW application

Figure 1 depicts a WWW application; a client initiates a request for a document which is available on some server. The server responds and sends the requested document back to the client.
Some of the key components of WWW will be explained in this chapter.

2.1 Clients

A WWW client, sometimes called a “browser” is a computer program which requests a document from a WWW server, and presents what it receives to its user. Often auxiliary or helper programs assist the client in presenting certain kinds of documents such as sound, picture, or video files. The most popular WWW client is Netscape, which comes in versions for Unix workstations running X-Windows, Personal Computers using Microsoft Windows, and Macintosh personal computer. Computers without graphical display capability can use text-only clients, such as Lynx, to browse WWW documents.

2.2 Servers

A WWW server is merely a relay system, receiving requests for WWW documents from clients and sending back the documents requested. WWW servers appeared first on UNIX hosts, but versions for Windows NT as well as other operating systems are now available. More sophisticated servers are able to pass requests for documents via the so-called Common Gateway Interface (CGI) to other applications (often databases), which return WWW-formatted information. These generated files will subsequently be transmitted by the server to the requesting client. A detailed discussion of WWW servers can be found in Section 3.

2.3 Documents

WWW documents consist of one or more files, and include text, picture, sound and sometimes video information. The text files include special codes that describe how it should be displayed and how it may be linked to other documents. These codes, which follow a standard called the

HyperText Mark-up Language (HTML) allow WWW client or browser applications to display the document with the appropriate visual formatting. A WWW document may also contain codes that direct the reader to other documents, which may be available on the same or on a different server. These other documents may again contain text, picture, sound or video information.

2.4 The Network

WWW is made possible because of the ubiquity of the Internet, a mega-network of computer systems which can be considered as the forerunner of the electronic super highway. The rules and formats for communication between WWW clients and servers are defined by application layer protocols, which in turn make use of underlying transport and network layer protocols (e.g. TCP/IP). One of the best known application layer protocols is the HyperText Transfer Protocol (HTTP). Depending on the information that is contained within the WWW document, the size of HTTP data units may be anything between 300 bytes and many megabytes. Until now most WWW users do not pay for the actual use of individual WWW servers, but instead pay a flat monthly rate to access the entire Internet. This is likely to be changed in the future, which implies that payment may become usage sensitive.

2.5 Uniform Resource Locators

Uniform Resource Locators (URLs) are identifiers for resources within the WWW. URLs consist of three parts:

- an identifier for the protocol that should be used to access the resource (e.g. http or ftp),
- the (DNS) name of the machine on which the resource resides,
- the precise name of the resource on that machine (often a file name)

Work is currently performed within the IETF to come up with a more universal identification scheme, called Uniform Resource Identifiers (URIs).

2.6 Virtual servers

Virtual domains is a technique to assign multiple IP addresses to a single physical interface. It reduces the cost of setting up servers by allowing multiple ‘virtual’ servers to coexist on a single piece of hardware. Although the servers are connected to the same physical network, they operate as if they belong to multiple domains. To realize virtual domains, the operating system should be modified to support multiple addresses for a single physical interface. This modification or ‘hack’ is already supported in many OS's, such as SunOS, SOLARIS, Linux and NT.

Also the alias technique provides the capability to assign multiple names to a single host. There is a difference however between the alias and the virtual domains technique: the alias technique should be used to assign multiple host names to the *same server*. An example of the alias technique is to use ‘wwwsnmp.cs.utwente.nl’ and ‘rose.cs.utwente.nl’ for the same server.

3 WWW servers

WWW applications can be decomposed into two conceptual layers:

- A layer responsible for the transparent *transfer* of WWW documents. This layer does not interpret the HTML formatting codes that may be contained within WWW documents. The operation of this layer is defined by the HTTP protocol.
- A layer responsible for the *storage* or *construction* (at the server side) and *formatting* (at the client side) of WWW documents. For this purpose, the client interprets the various HTML codes that are contained within the document. The precise way to transfer WWW documents is not significant within this layer.

As will be illustrated in Part III of this report, the idea to decompose the WWW into these two layers will be helpful to identify and structure possible WWW management information.

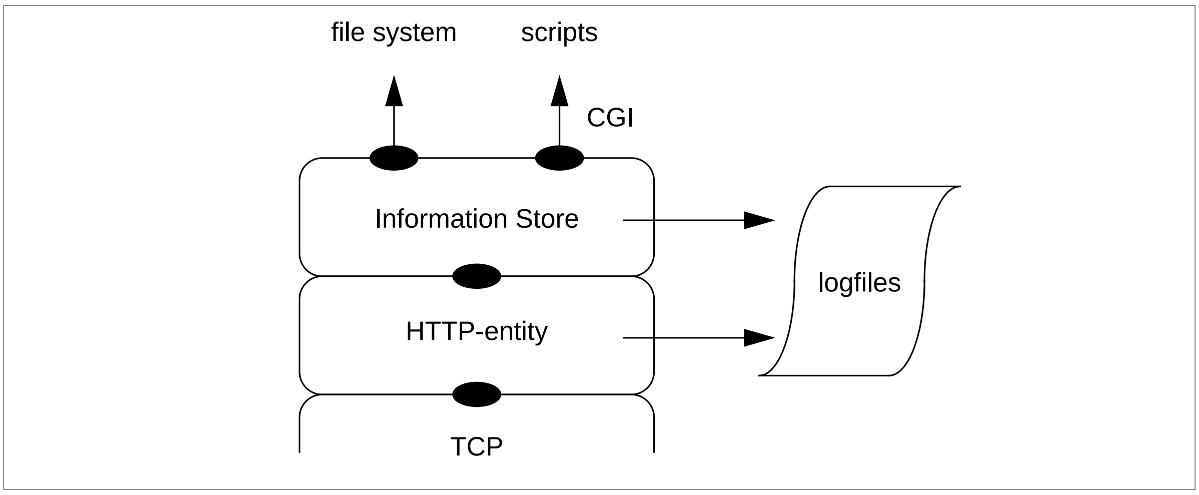


Figure 2: Structure of a WWW server.

Figure 2 shows the internal structure of a typical WWW server. The entity responsible for the storage of WWW documents is called the ‘Information Store’ (IS). This entity has access to the file systems and may as well communicate, via the Common Gateway Interface (CGI) and scripts, to external programs such as database systems. Information concerning the operation of the IS and the HTTP entities is stored within special logfiles.

3.1 File system

The file system is the place to store *static* WWW documents. Such documents are static, in the sense that they should be created *before* they can be requested by clients. Static documents consist of (a combination of) HTML formatted ascii files, image files, sound files and even video files. Home pages, articles and conference announcements are possible examples of static WWW documents.

The (network) file system is usually an integral part of the operating systems. Each WWW server gets his own *WWW root* within this file system. This concept of a WWW root is similar to the concept of a file system root, in the sense that all WWW documents should be stored in directories below the WWW root (see Figure 3).

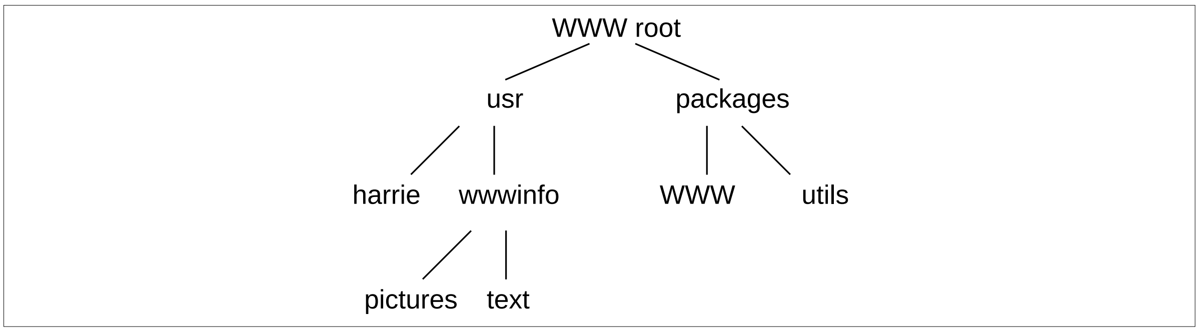


Figure 3: Example of a file system with WWW server root

3.2 CGI

The Common Gateway Interface (CGI) allows the server program to communicate with other programs, such as database systems. As opposed to static WWW documents, these external programs may dynamically create WWW documents after a request from a client has been received. This mechanism ensures that the provided information can always be up to date.

> ***Example:*** The CGI is used to pass queries to a system which includes all share prices of the stock market.

The external programs that dynamically produce WWW documents are usually running as separate processes on the host of the WWW server. Running CGI programs on your WWW server host is relatively insecure; clients who want to attack your system may 'hack their way' via CGI.

3.3 Logfiles

Most WWW servers maintain logfiles with information concerning client requests and server errors. Logfiles allow managers to obtain statistical information concerning the servers usage and to trace possible problems.

The *access logfile* includes the information of client requests. The format of this logfile, which is described in a de facto standard, is shown in Figure 4.

remotehost rfc931 authuser [date] "request" status bytes

- *remotehost* is the remote host name or IP address of the client.
- *rfc931* is the remote log name of the user.
- *authuser* is the name by which the user has authenticated himself.
- *date* is the date and time of the request.
- *"request"* is the request line (URL), exactly as it came from the client
- *status* is the returned status code to the client.
- *bytes* is the number of bytes transferred to the remote host.

Figure 4: Format of a logfile

Figure 5 gives an example of a logfile entry.

```
annex1s13.urc.tue.nl - - [06/Feb/1996:21:16:09 +0000] "GET  /vloot.html HTTP/1.0" 200 3321
```

Figure 5: Example of an access logfile entry

The *error logfile* includes the information of server, communication and information related errors (Figure 6). Unfortunately, the format of this type of logfile is not standardized.

Server related error:

```
[Thu Nov  2 16:00:37 1995] httpd: caught SIGTERM, shutting down
```

Communication related error:

```
[Sat Nov 25 18:33:58 1995] send timed out for case.cs.utwente.nl
```

Information related error:

```
[Tue Dec  5 11:05:57 1995] access to /home/misc/hazewink/APACHE/htdocs/ijzeil.gif
failed for cam027313.student.utwente.nl, reason: File does not exist
```

Figure 6: Example of error logfile entries

Part II

Network management technology

4 Protocol selection

The purpose of this chapter is to select which protocol should be used to manage WWW. It starts with an overview[1] of the protocols and techniques that are the potential candidates. These candidates are[2]:

- the Simple Network Management Protocol (SNMP)
- version 2 of SNMP
- the Common Management Information Protocol (CMIP)
- the Telecommunications Management Network (TMN)
- the protocols and techniques of the Network Management Forum
- the management protocol of the IEEE.

The overview sections are followed by a section that states the criteria on which the selection of the management protocol should be based. The chapter concludes with a recommendation concerning the protocol that should be used.

4.1 SNMP

In the second half of the previous decade the Internet Engineering Task Force (IETF) concluded that management of the Internet could no longer be provided on an ad-hoc basis. After some discussion, the IETF decided that it would be best to use the ‘Common Management’ protocol that was being developed as part of the OSI program [15]. To allow the OSI management protocol to run on top of a TCP/IP stack, minor modifications were necessary. The resulting protocol was called ‘Common Management Over TCP/IP’ (CMOT) [18].
Unfortunately, standardization of OSI management did not progress at great pace. To satisfy the immediate management needs of the Internet, the IETF therefore decided to introduce for the short term an extended version of the ‘Simple Gateway Monitoring Protocol’ (SGMP) [14]. This protocol was called the ‘Simple Network Management Protocol (SNMP) [17]; the idea was to replace this protocol as soon as OSI management standards were ready.
Apparently SNMP was the right solution at the right time. Already a few years its introduction, most datacommunication equipment could be managed via SNMP; SNMP had become the de facto management standard. The overwhelming success of SNMP urged the IETF to review its original plans and drop the support for OSI management: in 1992 it removed CMOT from the IETF standardization track.

The ideas behind SNMP are as follows:

- *All* systems connected to the network should be manageable with SNMP.
- The cost of adding network management to existing systems should be minimal.
- It should be relatively easy to extend the management capabilities of existing systems (by extending the Management Information Base).
- Network management must be robust. Even in case of failures, a small set of management capabilities must still be available.

SNMP allows a central manager to control the operation of many systems (Figure 7). The managed systems are called ‘agents’ and the behaviour of each of these systems may be monitored and modified via its ‘Management Information Base’ (MIB). Each MIB includes a large number of management variables [19]. These variables are expressed in terms of simple scalars and two-dimensional tables; as opposed to other management approaches it is not

1. Additional information can be found in [13].
2. If the project would be performed again, Web-based Management should also be considered.

possible to define structured variables. A further limitation is that SNMP can only operate on the individual elements of the table; it is not possible to operate on the table as a whole. As a result, SNMP management is performed at a relative low level of abstraction. In fact, SNMP may be seen as a 'remote debugging' facility. Its strengths are its simplicity, costs and ubiquity; it is not well suited however for complex management tasks.

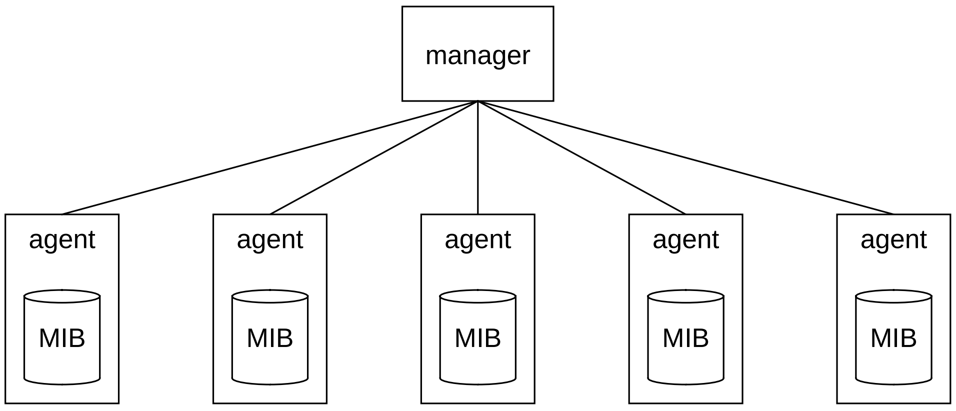


Figure 7: Manager - agent relationship

SNMP operations

Communication from the manager to the agent is performed in a confirmed way. The manager takes the initiative by sending one of the following PDUs: *GetRequest*, *GetNextRequest* or *SetRequest*. The *GetRequest* and *GetNextRequest* are used to retrieve management information from the agent, the *SetRequest* is used to store (or change) management information. After reception of one of these PDUs, the agent responds with a *Response* PDU (Figure 8). This PDU carries the requested information or indicates failure of the previous request.

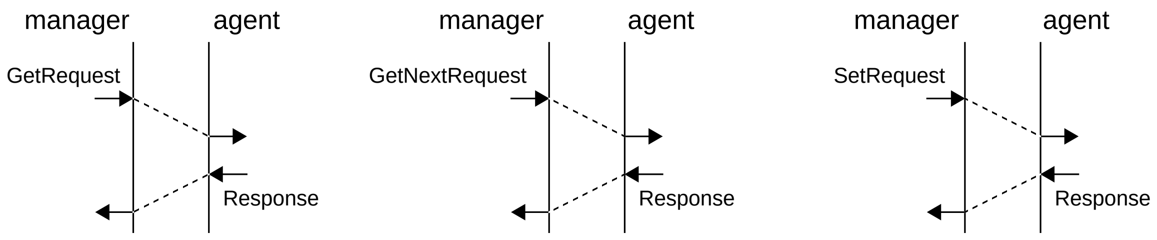


Figure 8: Manager takes the initiative

It is also possible that the agent takes the initiative. This happens in case the agent detects some extraordinary event, such as a re-initialization or a status change at one of its links. As a reaction, the agent sends a *Trap* PDU to the manager. Reception of the *Trap* is not confirmed (Figure 9).

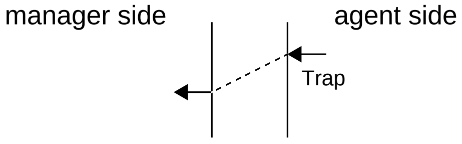


Figure 9: Agent takes the initiative

SNMP does not describe how a manager should relate the various *Get, Set* and *Trap* interactions. What to do after reception of a *Trap*, for example, is not defined by the standards.

4.2 SNMPv2

Since publication of the original SNMP protocol, several proposals have been presented to improve SNMP. In 1992 it was decided to collect these proposals and produce a new standard: SNMP version 2 (SNMPv2). As compared to the original version, version 2 offers better performance, better security and the possibility to build a hierarchy of managers. Next to these main improvements also a large number of small improvements were included.

SNMPv2 was first described in RFC 1441 - RFC 1452 [20]-[31] and became Proposed Standard in 1993. Soon after its publication several research groups started to implemented the protocol to obtain practical experience. It soon turned out that SNMPv2 was not as straightforward as expected. As in 1994 the IETF reopened the discussion to determine whether progression to Draft Standard was possible, many people complained about the complexity of the 'party based' administrative model. This model describes the 'parties' and 'contexts', which are exchanged between the manager and agent on behalf of the security functions (authentication, encryption and access control). Because of the ongoing criticism, two of the four original editors of SNMPv2 dropped their support for the 'party based' model and proposed to replace it by a much easier to understand 'user based' model.

In the three month that followed it was impossible to find sufficient support for this new model. In september 1995 the Network Management Area Director (NM-AD) therefore decided to freeze discussions on the complex security aspects of SNMPv2. Instead, it was decided to develop a version with the same simple (community based) security features of SNMPv1. This new version of SNMPv2 is described in RFC 1901-1908 [32]-[39]. At the start of this project it is to early to judge the support for this new version.

4.3 CMIP

The first standardization organization that addressed network management was the International Organization for Standardization (ISO). Around 1980 it formed a special working group to define OSI management. This group started to develop the 'OSI Management Framework' [7], which was intended to provide the architectural basis behind OSI management. Standardization of this framework and its associated management protocol, the 'Common Management Information Protocol' (CMIP) took many years[8]. In the mean time SNMP appeared and manufacturers of data communication equipment decided not to wait until completion of OSI management, but to implement SNMP instead.

Only recently manufacturers of *manager* systems (e.g. HP, SUN and BULL) started to implement CMIP. This support of CMIP may be motivated by the desire of these manufacturers to earn some money in the rapidly growing field of telecommunication management (CMIP oriented management systems are far more expensive than their SNMP oriented counterparts). In the datacom world, CMIP managers are only used in very large networks as 'top level managers'. This is because there are hardly any implementations of CMIP *agents* available; communication between manager and agent must therefore still be based upon SNMP.

From a technical point of view OSI management has advantages and disadvantages. One of the main advantages is the rich (object oriented) information model, which allows the creation of every possible managed object type (remember that SNMP allowed only creation of simple scalars and two dimensional tables). With CMIP it is also possible to directly operate on complex objects (SNMP required the manager to operate on each individual table element).

A disadvantage of OSI management is that many management functions should be performed by the managed systems. This hinders the ubiquity of OSI management; especially in case of relatively simple and inexpensive systems (PCs, workstations, bridges, modems etc.) addition of OSI management functions may be too complex and expensive.

4.4 TMN

The term TMN is introduced by the ITU-T (the former CCITT) as an abbreviation for ‘Telecommunications Management Network’. The concept of a TMN is defined by Recommendation M.3010 [2].

According to M.3010, "a TMN is conceptually a separate network that interfaces a telecommunications network at several different points". The relationship between a TMN and the telecommunication network that is managed, is shown in Figure 10. According to this figure, the interface points between the TMN and the telecommunication network are formed by exchanges and transmission systems (called ‘network elements’). For the purpose of management, these exchanges and transmission systems are connected via a ‘Data Communication Network’ (DCN) to one or more ‘Operations Systems’. The Operations Systems perform management functions, which may be carried out by human operators but also automatically. It is possible that a single management function will be performed by multiple Operations Systems. In this case, the DCN is used to exchange management information between the Operation Systems. The Data Communication Network is also used to connect Work Stations, which allow operators to interpret management information. Work Stations have man-machine interfaces, the definition of such interfaces fall outside the scope of TMN (Work Stations are therefore drawn at the border of the TMN).

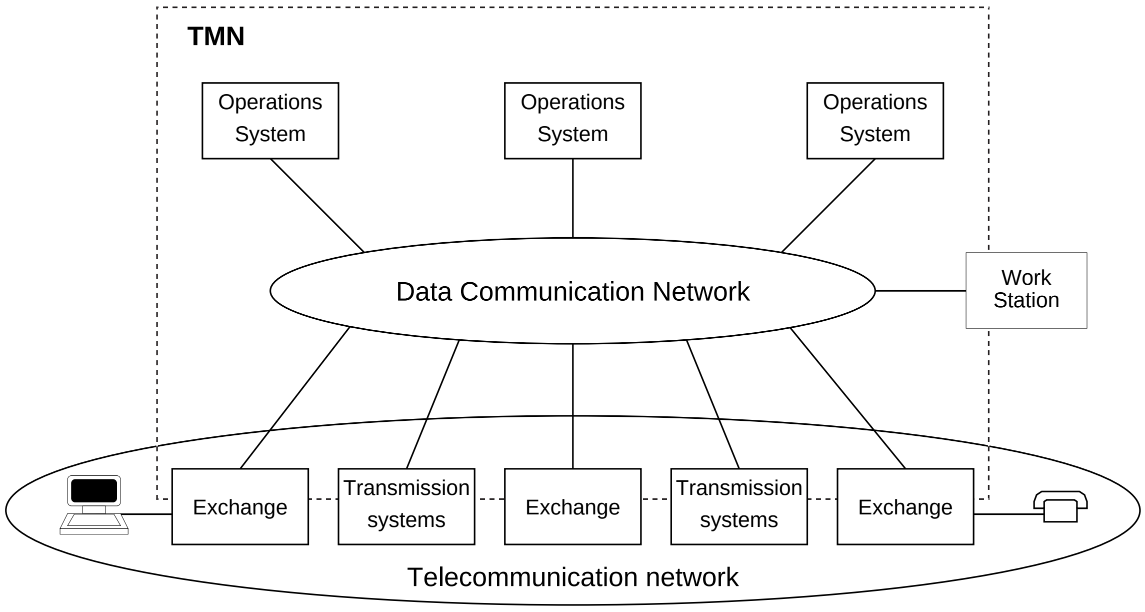


Figure 10: General relationship of a TMN to a telecommunication network

TMN should be seen as an attempt to align terminology and ideas. It provides an overview of management functions and structures at a high level of abstraction. Note that TMN does not introduce any new management protocols; the communication between operation systems and network elements TMN relies on the use of CMIP. For the purpose of this chapter it is therefore not necessary to consider TMN any further.

4.5 Network Management Forum

In 1988 the ‘OSI/Network Management Forum’ was formed to promote the rapid development, acceptance and implementation of ISO and CCITT management standards [5]. The Forum is a non-profit organization whose members are manufacturers, operating companies and research laboratories. After a few years the prefix ‘OSI’ was removed to indicate that the Forum had widened its scope to reference management standards from other sources.
Examples of such standards are:

- SNMP from the IETF.
- The ‘Distributed Management Environment’ (DME) [1] from the Open Software Foundation (OSF).
- The ‘Management Protocol API’ (XMP) and the ‘OSI-Abstract Data Manipulation API’ (XOM) from X/Open.
- The ‘Common Object Request Broker Architecture’ (CORBA) from the Object Management Group (OMG).

To organize its work, the NM Forum has defined the OMNI*Point*[1] program. This program comprises "a set of standards, implementation specifications, testing methods plus tools, object libraries that make possible the development of interoperable management systems and applications" [9][10]. Outside the telecom world the OMNI*Point* program was not very successful. Reasons for this lack of success seem to be:

- The goals set for DME were over-ambitious: it was built upon a number of false assumptions, it focused on *distributed management of objects* instead of *management of distributed objects*, object orientedness was seen as the solution for all problems and too many information models were being supported [42].
- XOM turned out to be more complex than expected [3]. In a number of implementations XOM was in fact replaced by a proprietary API.
- The delivery schedule envisaged for CORBA could not be met.

4.6 IEEE

The Institute of Electrical and Electronics Engineers (IEEE) is a professional organization which, amongst others, defines standards for Local and Metropolitan Area Networks (LANs and MANs). These standards are commonly known as the IEEE 802 standards. Some of these standards define how management should be performed in LAN and MAN environments (Figure 11).

Number	Title
IEEE 802.1B	LAN/WAN Management
IEEE 802.1E	System Load Protocol
IEEE 802.1F	Common Definitions and procedures for IEEE 802 Management Information

Figure 11: IEEE Management standards

The IEEE management standards are based upon the ISO CMIP standard. As opposed to ISO, IEEE does not use this protocol at application level (layer 7), but at data link level (layer 2). The name that is used for the IEEE approach, is Common Management Over LLC (CMOL). A

1. OMNI*Point* stands for Open Management Interoperability Point

problem with CMOL is that it is impossible to manage stations located at other sides of routers (routers, by definition, relay via layer 3). IEEE management is thus restricted to single (bridged) LANs or MANs; management of large internets is thus impossible.

4.7 Selection criteria

The management protocol that will be selected should satisfy the following criteria:

- the protocol should allow monitoring of the information servers, as well as the Internet to which these servers will be connected. Monitoring should be possible from a remote place, somewhere on the Internet.
- the protocol should be standardized, and not be proprietary technology.
- the protocol should be proven technology.
- the protocol should be accepted by the datacom market and available from several vendors.
- the protocol should support at least some limited level of security.
- implementation should be inexpensive.

4.8 Conclusions

Several of the protocols that were mentioned in the overview section do not satisfy the above criteria:

- TMN need not be considered because it does not define any management protocol itself (instead it references CMIP).
- CMOL can not be used over the Internet.
- The technologies of the network management forum can not be regarded as proven technology.
- The same is true for SNMPv2.

The real choice is therefore between (version 1 of) SNMP and CMIP. Although CMIP is potentially more powerful than SNMP, it is not (yet?) accepted by the datacom market and can hardly be considered as proven technology. Besides, the use of CMIP will be far more expensive because vendors ask relatively high prices (compared to SNMP) and various components (e.g. agents) may not be available elsewhere and need therefore be tailor made for this project.

The conclusion is therefore to use SNMP. In the mean time progression of SNMPv2 should be monitored. If this new version becomes a success, the decision to use SNMP version 1 may be reconsidered (it is assumed the new version will be capable to operate with version 1).

5 Evaluation of Network Management Products

This chapter presents an overview of the Public Domain (PD) network management products[1] that can be used to develop WWW management agents. To select an appropriate network management product, several requirements are identified (Section 5.1); each network management product will be evaluated with respect to these requirements (Section 5.2). The chapter concludes with a recommended network management product (Section 5.3).

5.1 Requirements

To guide the selection of a network management product for the development of an agent, several requirements are needed. The following requirements were identified during the meeting in Ispra (27 July 1995). The most important requirements were:

- The agent must at least support the SNMPv1 protocol.
- The agent must run under the Solaris operating system
- The agent must preferably implement the MIB-II [19]. The MIB-II provides information on several Internet protocols, for instance, Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP).

Some requirements a bit less (but still) important were:

- Extensible agent technology will be preferable.
- The agent must preferably be freely distributed among customers on the Internet, so it is attractive whenever the agent implementation could run on as many systems as possible.

To simplify the implementation of the WWW-MIB and the WWW agent, some additional desirable properties were identified by the project partner:

- The product must include tools for easy MIB implementation. Such a tool can be a MIB compiler or an SNMP implementation capable of auto-loading a MIB specification. During the loading process, an internal MIB representation will be build corresponding to the hierarchy of the MIB specification. The internal MIB representation is suited to store the entire MIB. Usually, such a MIB specification is written in the Abstract Syntax Notation One (ASN.1) language.
- To implement an WWW agent, it seems attractive to modify an already existing agent. The source code of the agent must therefore be available.

5.2 Software Products

On the Internet, several interesting network management products can be found. The most important of these products are summarized below. The functionality of every product is globally described and then the product is evaluated against the requirements mentioned before.

5.2.1 The CMU-SNMP package

The CMU-SNMP package is developed at the Carnegie Mellon University (USA). The package was build primarily to improve the world-wide acceptance of the SNMP protocol. The main developer of this package is Steve Waldbusser, one of the architects of SNMPv2.

1. An overview of commercial management products has been performed by ESYS Ltd. and will not be discussed in this report.

The main part of the package consists of an SNMPv1/v2 protocol stack. This stack is available to an application programmer via an C programming interface. The protocol stack is capable of auto-loading a MIB written in the ASN.1 language. Around this stack, a complete MIB-II compliant agent has been developed. This agent is probably the most widely and frequently used SNMP agent. The implementation seems to be very stable during the past years.
Besides the aforementioned ANSI-C interface, a Tool Command Language (TCL) interface and a PERL interface is also available. Unfortunately, the supported TCL version is outdated, but the PERL interface is up-to-date.
The entire CMU-SNMP package is in source code available. The package is poorly documented: just some rudimentary manual pages are included in the package. No extensible agent technology is supported. The package contains several manager programs, which are very suitable for testing purposes.
The CMU-SNMP package is ported to the Solaris operating system.

5.2.2 The ISODE package

The ISODE software is primarily intended for the development and testing of new Internet protocols. The SNMPv1/v2 part of the ISODE package is primarily build by Marshall T. Rose, who is probably the most important architect of the SNMP framework.
The SNMPv1/v2 protocol stack in the ISODE package are accessible via an ANSI-C and TCL application programming interface. A MIB-II compliant agent completes the SNMP implementation. The entire ISODE package is in source code available. The package is reasonably documented, but the enormous disksize needed for this package (around 100 megabytes) can be a major drawback. Another drawback is the implementation structure of the SNMP agent, which is not always very clear.
The ISODE SNMP implementation seems to be very stable. A MIB compiler is included in the package, and the SNMP implementation supports the building of extensible agents. Unfortunately, the extensible agent technology (SMUX) is outdated and not supported any more. The ISODE software is available for the Solaris operating system.

5.2.3 The SCOTTY software

The SCOTTY software, written at the Technical University of Braunschweig in Germany, consists of several individual tools; the SCOTTY tool, the BONES network configuration database and the TKINED graphical network editor.
The SCOTTY tool provides a TCL interface to several internet protocol stacks, for instance, SNMPv1/v2, UDP, TCP, ICMP and the HyperText Transfer Protocol (HTTP). The TCL interface supports the latest version of the TCL/TK language. A MIB, specified in the ASN.1 language, is auto-loaded in the SCOTTY tool and the internal corresponding MIB representation can immediately be used to store MIB information. The internal representation in the SCOTTY tool is accessible via the aforementioned TCL interface.
The documentation could have been better, but the software is stable. A TCL script of an simple agent is included for demonstration purposes. All tools are in source code form available. The SCOTTY is one of the newer network management tools on the Internet. No extensible agent technology is supported. A Solaris implementation is available.

5.2.4 The Tricklet software

The purpose of the Tricklet software, developed at the Technical University of Delft in the Netherlands, is to provide tools for performability and manageability of Local Area Networks (LANs).
The Tricklet package comprises a SNMPv1 protocol stack. The included agent implements the Remote Monitoring (RMON) MIB. The SNMP protocol stack can be accessed via a PERL language interface. This interface does not support the latest version of the PERL language. The Tricklet software is well documented and especially the agent implementation seems to be widely applied. No extensible agent technology is supported and the Tricklet software is available for Solaris.

5.2.5 The SNMP Development kit

The SNMP software developed at the Massachusetts Institute of Technology is very well structured. Although the MIT software includes partly MIB-II support, unfortunately only SNMPv1 is implemented in the development kit. The MIT software is in source code available, but lacks any documentation. The software is not ported to the Solaris operating system. No extensible agent technology is available.

5.2.6 The UT-SNMP package

The UT-SNMP software is one of the results of the UT-SNMP project at the University of Twente in the Netherlands. One of the goals of the project was to implement the entire SNMPv2 framework based on the Internet Request For Comment documents 1441-1452.
The UT-SNMP software provides primarily a SNMPv2 protocol stack. Unfortunately, the latest UT-SNMP version lacks the support of SNMPv1. Several applications, a simple agent and manager, are added for demonstration purposes.
The SNMP protocol stack is accessible via an ANSI-C and TCL interface. The documentation is a bit outdated. A MIB compiler for ASN.1 specifications is available. The UT-SNMP package runs under the Solaris operating system and supports extensible agent technology. The package is in source code available.

5.2.7 The WILMA toolkit

This WILMA toolkit, developed at the Technical University of Munich, provides a SNMP based set of tools, such as a MIB compiler, MIB browser, and several demonstration SNMP agents. The demonstration agents contain several pre-defined MIB implementations, for instance the RMON-MIB, SNMP-MIB and the HOST-MIB. Extensible agent technology is not included in the toolkit.
The toolkit supports version 1 and version 2 of the SNMP protocol. The SNMPv2 is new and seems not very stable. The protocol stack is accessible via an C application programming interface. The documentation is very clear. The toolkit is ported to the Solaris operating system. The source code of the WILMA toolkit is freely available.

5.2.8 Summary of PD products

The table below shows all public domain products, as well as the email address, the ftp location and the universal resource locator (url) address for additional information.

Name of product	Organisation	Email address / Url address / Ftp location
CMU-SNMP	Carnegie Mellon University (USA)	Email address: Url address: Ftp location: ftp://lancaster.andrew.cmu.edu/pub/snmp-dist/
ISODE		Email address: ISODE-SNMPv2@ida.liu.se Url: Ftp location: ftp:/ftp.ics.uci.edu/mrose/
RMON	University of Delft (NL)	Email address: btng@dnpap.et.tudelft.nl Url address: http://dnpap.et.tudelft.nl/DNPAP/Software/software.html Ftp location: ftp://dnpap.et.tudelft.nl/pub/btng
SCOTTY	University of Braunschweig (DE)	Email address: tkined@ibr.cs.tu-bs.de Url address: http://www.cs.tu-bs.de/ibr/projects/nm/scotty/ Ftp location: ftp://ftp.ibr.cs.tu-bs.de/pub/local/tkined/
MIT	Massachusetts Inst. of T. (USA)	Email address: snmp-dk@allspice.lcs.mit.edu Url: Ftp: ftp://ptt.lcs.mit.edu/pub/snmp/
UT-SNMP	University of Twente (NL)	Email address: snmp@cs.uwtente.nl Url address: http://wwwsnmp.cs.utwente.nl/ Ftp location: ftp://ftp.cs.utwente.nl/pub/src/snmp/software/
WILMA	University of Munich (DE)	Email address: wilma-list@ldv.e-technik.tu-muenchen.de Url address: file://ftp.ldv.e-technik.tu-muenchen.de:/dist/WILMA/ Ftp location: ftp://ftp.ldv.e-technik.tu-muenchen.de/dist/WILMA/

The following table summarizes some of the product specific information.

Name of Product	GUI	PI	PL	Sp	Sr	T	V
CMU-SNMP 2.1	X11 Windows	Ansi-C TCL PERL	Solaris HPUX LINUX	no	yes	MIB-II agent and manager SNMP stack	1/2
ISODE 8.0	ascii	ANSI-C TCL	Solaris AIX HPUX	no	yes	MIB II agent SNMP stack MIB compiler	1/2
Tricklet 3.1	X11 Windows	ANSI-C PERL	Solaris LINUX OS2	yes	yes	RMON agent SNMP stack	1
SCOTTY 2.0	X11 Windows	TCL	Solaris LINUX HPUX and many others	yes	yes	SNMP stack Simple agents	1/2
MIT 1.1	ascii	ANSI-C		no	?	SNMP stack	1
UT-SNMP 4.0	ascii	ANSI-C TCL	Solaris HPUX	yes	yes	SNMP stack MIB compiler Simple agent and manager	2
Wilma 2.0	X11 Windows	ANSI-C	HPUX/ Solaris/LINUX ...	yes	yes	SNMP stack MIB compiler agents and manager	1/2

Legend:
- GUI: Graphical User Interface
- PI: Programming Language Interface
- PL: Running on PLatforms
- Sp: Support
- Sr: Sources available
- T: Tools
- V: SNMP version 1 or SNMP version 2 support

5.3 Conclusions

Two of the most important requirements, the SNMPv1 - and MIB-II requirements, selects two public domain products; the ISODE - and the CMU-SNMP package. The Solaris requirement is supported by both packages.
Both products support the easy implementation of MIBS. Although the ISODE package includes a MIB compiler, the CMU-SNMP package provides the build-in facility for auto-loading MIBS.
Both packages provide a stable SNMP agent. It seems that the CMU-SNMP agent is more preferable, due to the more clear implementation structure. This agent implementation makes the incorporation of a new WWW-MIB easier and more straight forward.
The enormous disksize of the ISODE package is another significant drawback.
The concluding recommendation is to apply the *CMU-SNMP package* for the development of a SNMP agent with a WWW-MIB implemented.[1]

1. After completion of the ‘Overview of commercial network management products’, it was decided to use SNMP-Research’s EMANATE package; primarily because of its extensible agent technology capabilities.

Part III

Definition of the WWW MIBs

6 Introduction to the WWW MIBs

To monitor the utilisation and performance of WWW servers, a number of Management Information Bases (MIBs) must be defined by the project.
This part of the report describes the design and structure of these MIBs. Section 6.1 identifies the user requirements and Section 7 shows the basic design. In this design three different MIBs will be distinguished: a HyperText Transfer Protocol (HTTP) MIB, a Retrieval Service (RS) MIB and an Information Store (IS) MIB). These MIBs will be presented in Sections 8 through 10.

6.1 User Requirements

To guide the development of WWW MIB modules, the following user requirements were identified:

- The MIB modules must include management information for the monitoring of WWW servers. Modules for WWW clients may be considered too, but are not of prime concern.
- The MIB modules must also include management information to monitor the performance of the transport network that is used by the various WWW systems.
- The structure of the MIB modules should be generic, which means that their basic structure should also be applicable in case alternative protocols and information storage systems will be used.
- The WWW server MIB module must include management information concerning possible support applications that are used to dynamically generate information (for example a database manager which interacts with the WWW server via the CGI interface).

7 WWW MIB design.

World Wide Web (WWW) enables the provision and collection of hypertext linked information via the Internet. Anyone who wants to provide information can connect to the Internet and makes his or her information available via WWW; anyone who wants to collect information, can also connect to the Internet and thus the WWW.
The enormous growth of the Internet and the World Wide Web makes management an important albeit difficult task. A prerequisit for good management is that management information should be well-defined and structured in a logical way. Within our project, the management information has been structured into a number of independent modules, each having a well-defined purpose.

7.1 Application structure

The distributed WWW application can be decomposed into two parts:
- An information dependant part, which is responsible for the provision and interpretation of hypertext linked information.
- An information independent part, which is responsible for the transfer of information and does not interpret the information. Within the project this part is called the Retrieval Service.

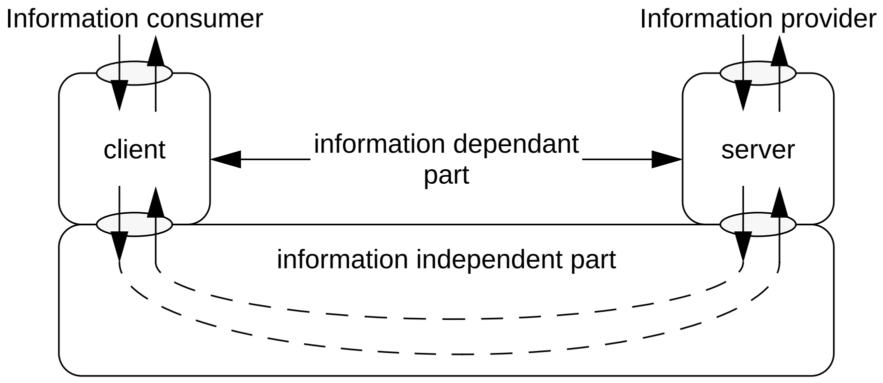


Figure 12: application parts and its underlying service.

This decomposition is shown in figure Figure 12. Because of the request/response paradigm of the WWW application, two different application parts are recognised; the server part, which is the interface towards the information provider and client part, which is the interface towards the information consumer.

7.2 Retrieval Service

The Retrieval Service (RS) provides a connectionless service to its users (the client and server entities). In the specific case of World Wide Web, the Retrieval Service can be decomposed into a HyperText Transfer Protocol (HTTP) on top of an Transmission Control Protocol (TCP) / Internet Protocol (IP) stack. This is shown in Figure 13.
The WWW designers preferred TCP above the User Datagram Protocol (UDP), because TCP is able to guarantee error free information exchange. As a consequence, the HTTP entities do not need to include recovery procedures.

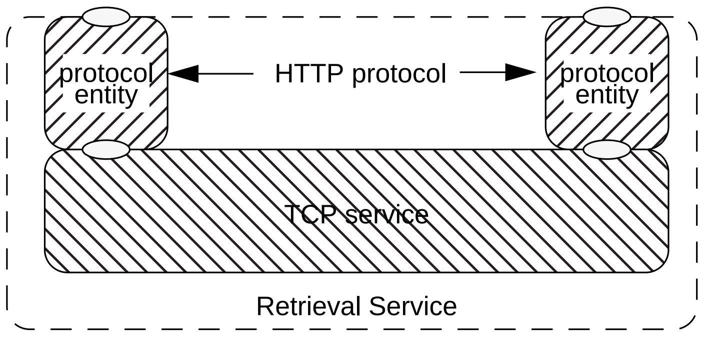


Figure 13: HTTP and its underlying service (TCP).

7.3 Management Information Bases

According to the user requirements of Section 6.1, the Information provider part as well as the Information transfer part (the Retrieval Service) that were identified during the previous decomposition should be manageable. There is no strict requirement however that the Information consumer part should be manageable.
To enable such management, MIBs should be associated with each of these parts (Figure 14).

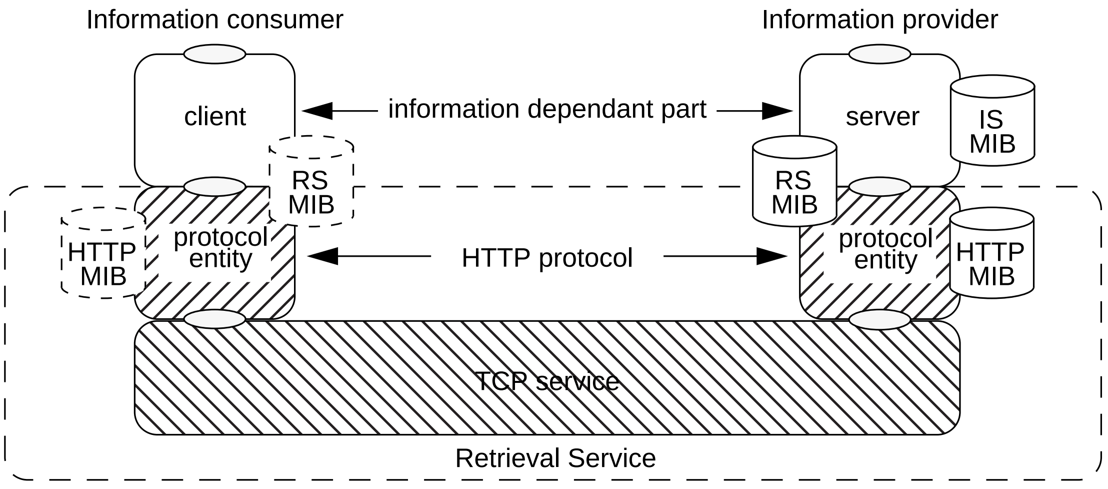


Figure 14: WWW MIB distributed over the WWW application

Three kind of MIBs have been identified:

- An Information Storage (IS) MIB, which contains managed objects concerning the documents (the information) that are provided by the WWW server. An example of such an object is a table that indicates how often each individual document has been accessed. This MIB, which need not be available at the client side, will be discussed in Section 10.
- A Retrieval Storage (RS) MIB, which contains managed objects that monitor the transfer of documents. This MIB, which will be discussed in Section 9, contains for example objects that count the number of service primitives and that measure the provided Quality of Service (QoS). Although this MIB must in principle be implemented at each Retrieval Service Access Point, it may in practice be sufficient to implement this MIB only at the server side.
- A HyperText Transfer Protocol (HTTP) MIB, which contains information to control the operation of the HTTP protocol. An example of such information is a PDU (=message) counter. Due to the symmetric nature of the HTTP protocol, the same MIB structure can be used for the client as well as the server side. The HTTP MIB will be discussed in Section 8.

Management of the lower level TCP/IP protocol stack does not require the definition of new MIBs, since the existing MIB-II [19] can be used for this purpose.

8 HyperText Transfer Protocol MIB

Although HTTP entities can act in a number of different roles, e.g. a client, server or proxy role, their fundamental behaviour is the same. It is therefore possible to define a single MIB structure, and have a special variable that indicates the role each entity is performing.
Figure 15 shows the basic structure of the MIB; the most important groups (httpSystem, httpStatistics and httpTimeOuts) will be introduced below.

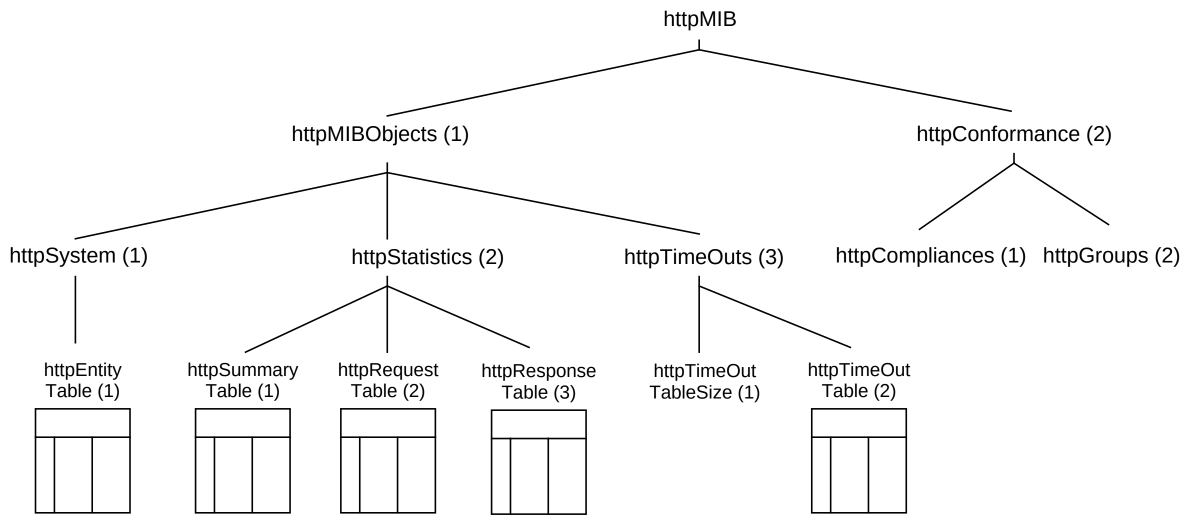


Figure 15: HTTP MIB structure.

8.1 System group

The System group consists of a single table: the httpEntityTable. Entity specific information is organised as a table, since more than one HTTP entity may be running on a single host machine. The table includes information concerning the possible use of virtual domains (Section 2.6) and is indexed with a unique entity identification number, which is also used as index in the other tables of this MIB.
For each entity the following information is provided:
- The kind of protocol implemented by the entity. This value should be derived from the 'list of assigned numbers'; in the case of HTTP this number is 80.
- A brief description of the entity and an email address of the responsible person.
- The version of the protocol that is implemented, the producer of the software, the software release version and (optionally) an object identifier for the producer.
- Configuration information, such as the address of the host machine (both in text form and as an IP address), the port in use, the time of last initialisation, and whether the entity is a client, server, proxy, or a caching proxy.

8.2 Statistics group.

The Statistics group provides information concerning the PDUs received and transmitted by the entity. The group consists of the httpSummaryTable, the httpRequestTable, and the httpResponseTable.

The SummaryTable contains for each HTTP entity a set of counters which provide a quick summary of the number of requests received, bytes transmitted, and so on. The SummaryTable also holds counters for requests and responses which have been discarded or received in error. Note that certain variables are redundant with respect to the Request and Response tables, they have been included however to reduce network traffic.

The Request and Response tables provide detailed information, broken down by entity as well as by type of Request / Response. Each Request / Response type has a separate table entry, giving the total count of the number sent / received, as well as a time-stamp to denote the last interaction.
To allow easy addition of new PDU types, for instance in case a new version of the HTTP protocol is defined, the information has been organized as tables. One of the indexes of the table is the PDU type. In case the agent detects a new PDU type, it should automatically extend the table with a new row to store the information concerning this new PDU type.

8.3 TimeOuts group.

The last group in the HTTP module contains time-out information for each of the HTTP entities. The information is presented in the form of a table, and may be resized by the Network Management system. It contains the address of the remote party involved, as well as the time at which the time-out occurred. Since the HTTP protocol specification does not define time-outs, discussions are needed whether this table is really useful.

9 Retrieval Service MIB

The Retrieval Service MIB contains high level information, such as service primitive statistics and Quality of Service (QoS) figures, concerning the transfer of documents. It abstracts from the specific protocols that provide this service, and has therefore some similarities with the service management MIBs that are defined by TMN [2]. Note however that, as opposed to the TMN service management MIBs, the RS-MIB provides management information concerning a single Service Access Point (SAP) only. To obtain an overview of the entire service, the manager should collect and combine the information from all RS-MIBs.

Within the context of this project it is of course not feasible to implement the RS-MIB in all systems (servers as well as clients). Still it is expected that useful management information may already be obtained if the MIB is implemented only within server systems.

The structure of the RS-MIB is shown in Figure 16; its main groups (rsStatistics and rsQoS) will be discussed in the following subsections.

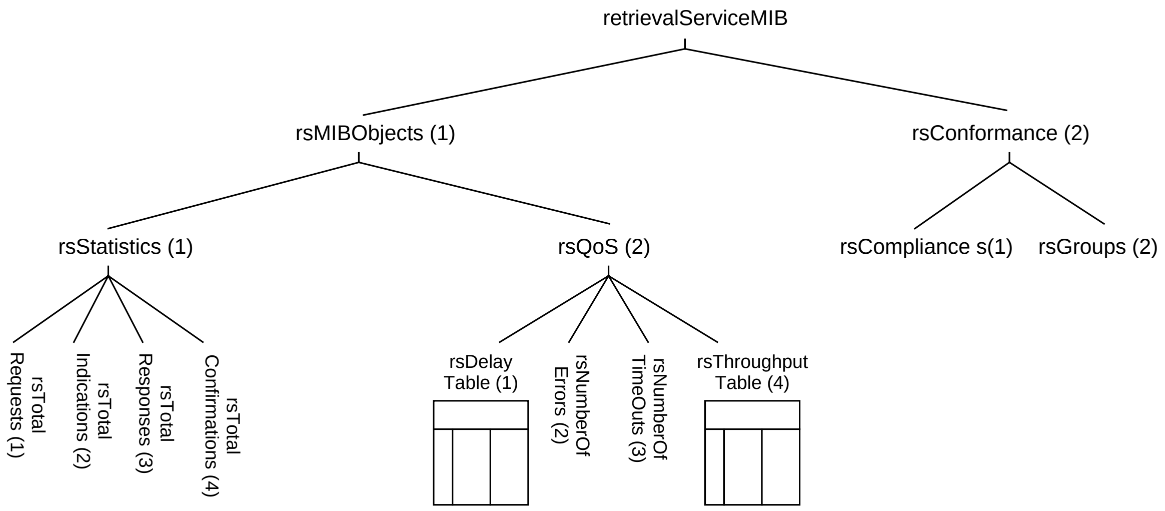


Figure 16: The Retrieval Service MIB structure.

9.1 Statistics group.

Interaction with the Retrieval Service takes place by means of service primitives. The statistics group contains objects that count, at a single Service Access Point (SAP), the occurrence of these service primitives.
Two possible solutions exist to define these counters:

- The total numbers of service primitives are counted by type. This requires for each service primitive type a dedicated managed object.
- The total number of service primitives are counted by type *and* remote host. This requires the introduction of a table, indexed by the service primitive type as well as the IpAddress of the remote host. Note that the remote host is a parameter of the service primitive.

For reasons of simplicity it was decided to implement the first solution.

9.2 Quality of Service group

The Quality of Service QoS group contains network management information about the quality of the document transfer. The information in this group provides the network manager with high level visibility of the performance of the underlying network. The recognised QoS values are: the transport delay, the number of errors, the number of time-outs and the throughput.

Transport delay values are stored within a table, which is indexed by the source and destination addresses of the communicating systems. Figure 17 shows an example.

source(1)	destination(2)	delay (3)
x.x.x.x	y.y.y.y	13
x.x.x.x	z.z.z.z	25
y.y.y.y	x.x.x.x	18

Figure 17: Example of a delayTable

The total round trip delay can be computed by the Network Management System (NMS). If, for example, the NMS wants to know the round-trip delay from host x.x.x.x to host y.y.y.y, it should retrieve and subsequently add the variables ‘delayTable.1.3.x.x.x.x.y.y.y.y’ and ‘delayTable.1.3.y.y.y.y.x.x.x.x’. In the example of Figure 17, the round-trip delay is 13 + 18 = 31.

The instrumentation to obtain QoS values out of the system may not be easy to implement. One of the ways to obtain at least QoS *estimates*, is to use ‘ping’ and the ‘echo’ protocol. It should be noted however that ‘ping’ and ‘echo’ operate at network respectively transport level, and not at HTTP service level. The obtained QoS estimates may therefore not be correct.

10 Information Store MIB

The Information Store (IS) MIB specifies the network management information pertaining to WWW servers. This section discusses the four major IS groups: isGeneral, isAccess, isErrors, and isDocuments. The structure of the IS MIB is shown in Figure 18.

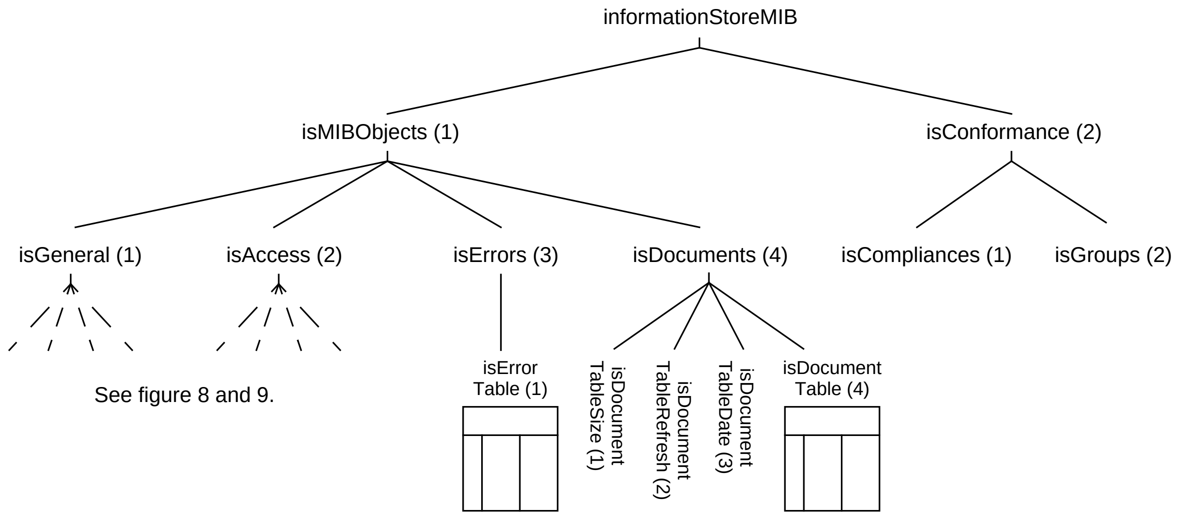


Figure 18: The Information Store MIB structure.

10.1 General Group

The isGeneral group (Figure 19) contains overall administrative data concerning the identity of the IS. It includes simple variables which holds the server's name, the organisation that operates the server, the contact address of the responsible person, the time the server was last initialized and a variable for the supported media types (e.g. text, pictures, sound and movies). Note that a variable to indicate the location of the server is missing; such variable is already available within the system group of the MIB-II and should not be duplicated.

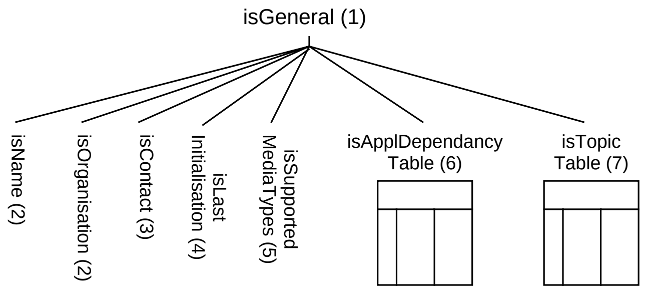

isGeneral (1)
isName (2)
isOrganisation (2)
isContact (3)
isLast Initialisation (4)
isSupported MediaTypes (5)
isApplDependancy Table (6)
isTopic Table (7)

Figure 19: The information store general group

The isGeneral group also includes two tables: isApplDependancy and isTopic. The last one indicates the kind of information that is provided by this server. The isApplDependancy table, which is needed in case the server operates as a proxy server, shows which applications (e.g. an Oracle database) are used to generate documents. It includes the application's name, version number, uptime, operational status and errors.

For the purpose of the project it was decided to initialize the isApplDependancy table by reading a configuration file. This file, which should be maintained off-line, contains the names of all associated applications. The status of these applications can be checked dynamically however, by using the 'process status' command of the UNIX environment.

In fact the isApplDependancy table provides the same kind of information as the system application (SYSAPPL) MIB, for which standardization has also been started within the IETF. In the future the applicationTable may therefore be replaced by the SYSAPPL-MIB.

10.2 Access group

To inform the manager about server usage, the access group (Figure 20) provides the following information:
- general summary information,
- domain specific information,
- document usage information, for the last N days,
- information concerning the most frequent and the most recent users.

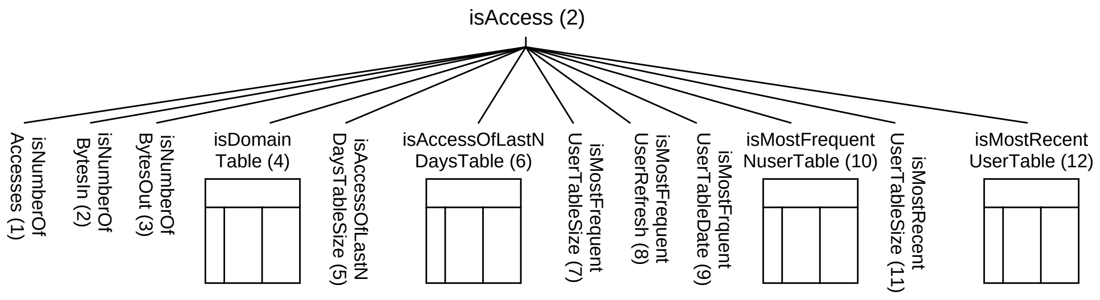

isAccess (2)
isNumberOf Accesses (1)
isNumberOf BytesIn (2)
isNumberOf BytesOut (3)
isDomain Table (4)
isAccessOfLastN DaysTableSize (5)
isAccessOfLastN DaysTable (6)
isMostFrequent UserTableSize (7)
isMostFrequent UserRefresh (8)
isMostFrquent UserTableDate (9)
isMostFrequent NuserTable (10)
isMostRecent UserTableSize (11)
isMostRecent UserTable (12)

Figure 20: The information store access group

The following general summary information is provided: the total number of accesses (isNumberOfAccesses), the total number of bytes received (isNumberOfBytesIn) and the total number of bytes transmitted (isNumberOfBytesOut). Although the manager could compute these three figures from other MIB information, they were included to avoid the large amount of traffic that must otherwise be exchanged for this computation.

Domain specific information is provided by the isDomainTable, which breaks down the accesses to the IS by Internet domain. The table allows the manager to determine how often the server has been accessed from each individual domain. This information helps the manager to decide whether and where mirror servers may be useful to reduce network traffic and server load.

The isAccessOfLastNDays Table shows how often the server has been accessed during the last couple of days. It allows the manager to determine whether server usage increases, and if additional server capacity is needed in the near future.

To understand which users have visited the IS, an isMostRecentUserTable and an isMostFrequentUserTable have been defined. Both tables are typical ‘TopN’ tables. The size of these tables, as well as the size of the isAccessOfLastNDays table, can be determined by the manager and may not exceed 128.

Figure 21 shows an example of the isMostRecentUserTable. Since updating this table may be expensive in memory as well as processing time, it will not be updated automatically, but only after an explicit request is received from the manager. A special ‘refresh’ variable has been included for this purpose. As a result of this request, the agent recalculates the table and updates the time-stamp variable that indicates the date and time when the table was last refreshed.

mfIndex(1)	mfUserName(2)	mfNumberOfAccesses(3)
1	ppp_7.mad.servicom.es	120
2	dijkstra.cs.utwente.nl	87
3	jipdmac.rb.noda.sut	36

Figure 21: Example of an isMostFrequentNUsersTable

The isMostRecentUsers table provides information on which users are the last ones who accessed the IS. The name of this table may be somewhat misleading, since it does not unveil the real identities of the (human) users that access the server (such information is not conveyed by the WWW protocols), but the names of the machines from which the users operate. These names should preferably be in DNS format, but IP addresses are allowed too. The table also contains the time of the last access, as well as the document that was accessed. Figure 22 shows an example of such table in which the isMostRecentNUsersTableSize is equal to 3. The most recent user is ppp_7.mad.servicom.es, who retrieved the document 'index.html' and immediately afterwards 'worldmap.xbm'.

mrIndex(1)	mrUserName(2)	mrlastTime(3)	mrDocument(4)
1	ppp_7.mad.servicom.es	07 cc 01 13 0c 0f 27 00	worldmap.xbm
2	ppp_7.mad.servicom.es	07 cc 01 13 0c 0f 26 00	index.html
3	jipdmac.rb.noda.sut	07 cc 01 13 0c 0d 10 00	iso/mngt-arti.html

Figure 22: Example of an isMostRecentNUsersTable

10.3 Error Group

The error group contains network management information about errors which occurred during IS access. This group does not include possible network errors, but only errors that relate to information access (e.g. document not available). The group is defined as a table with an error description, the number of occurrences of that error as well as the time at which the error last occurred. The reason to define this information as a table, is that extending tables with new error types can be performed without changing the MIB module definition.

10.4 Document Group

The document group contains network management information about the documents that are provided by the Information Store. The information is kept in a table, showing the document's name, access rights[1], size, type, and other features such as the number of accesses, time of last update, and any associated errors. The isDocumentTable has the same implementation constraints as the isMostFrequentUserTable, and is handled in a similar fashion with TableSize, TableRefresh, and TableData variables. It should be noted that (in the current implementation) the isDocumentTable is populated from the log files, and thus documents which have not been accessed will not appear in the table.

1. It should be noted that these access rights refer to the UNIX filesystem, and not to WWW. It is therefore not completely clear if variables that show these access rights are really useful.

Part IV

Implementation of the management agent

11 Introduction to the WWW agent

This part of the report describes how the agent was built. Subsection 11.1 summarizes the user requirement to which the implementation has to comply. Section 12 discusses the implementation environment; it provides a tutorial of the EMANATE package which was used to develop and implement an extensible agent. EMANATE is a kit to develop sub-agents, and is made by SNMP Research. Section 13 presents the design of the WWW sub-agent; Section 14 provides a detailed description of the sub-agent's functional parts.

11.1 User requirements

The project agreed on the following user requirements:

- Network management of WWW servers will be done with the Simple Network Management Protocol.
- Because standardisation of SNMP version 2 is not yet complete, it was agreed to use SNMP version 1.
- Extensible agent technology is preferred.
- The agent has to support next to the WWW MIB also MIB-II.
- Different WWW servers should be supported (the solution should thus be generic).
- WWW server software should not be altered to include management instrumentation.
- An attempt should be made to support a broad range of alternative protocols, such as gopher and ftp.

12 Implementation environment

The project decided to use the EMANATE package, made by SNMP Research Inc. (Knoxville, Tennessee, U.S.A). EMANATE (Enhanced MANagement Agent Through Extensions) supports SNMP version 1, as well as an early (and therefore not standardized) variant of SNMP version 2. The package supports extensible agent technology, which means that it uses a master agent to which sub-agents can be connected (Figure 23). Although the master and sub-agents are implemented as separate UNIX processes, their external behaviour is the same as if they were implemented as a monolithic SNMP agent.

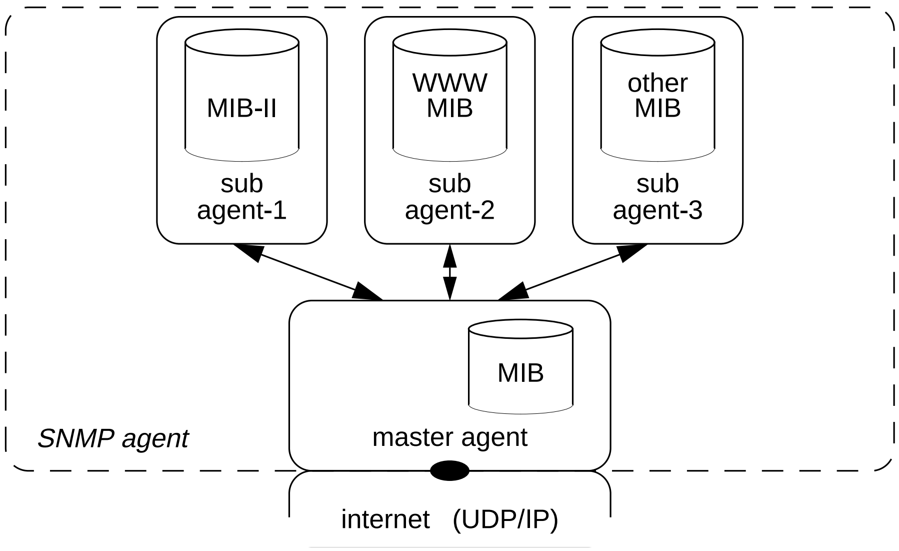


Figure 23: The EMANATE architecture.

The master agent includes a small MIB, which contains the variables for the MIB-II system and SNMP group. All other MIB variables are contained within the various sub-agents. For several standard MIBs, for example the MIB-II, ready to run implementations can be purchased from SNMP Research. To implement other MIBs (e.g. our WWW MIBs), SNMP Research offers an easy to use Sub-Agent Development Kit (SADK). The WWW MIBs that were discussed in the previous part of this report, have been implemented with this kit; Section 12 explains how this was done.

EMANATE allows duplication of MIB variables, which means that different sub-agents may contain variables with the same name. For this purpose EMANATE supports the following mechanism. If a sub-agent registers a variable, the master agent checks whether the variable’s name is already in use. If the name has not yet been used, EMANATE will pass subsequent requests for this variable to the sub-agent that registered the variable. If the name has already been used, the newly registered variable will be ignored. To allow communication between the master and sub-agents, a proprietary ‘protocol’ (a standard for such a ‘protocol’ is under development by the IETF) has been defined by EMANATE. It is also possible that sub-agents communicate among each other, but always in an indirect way via the master agent. EMANATE supports dynamic addition and deletion of sub-agents, without the need to recompile, start or stop the master agent (plug and play concept).

12.1 Sub Agent Development Kit.

To add a new MIB, the EMANATE package comes with a Sub Agent Development Kit (SADK). This kit consists of various tools and libraries with which sub-agents can be implemented.

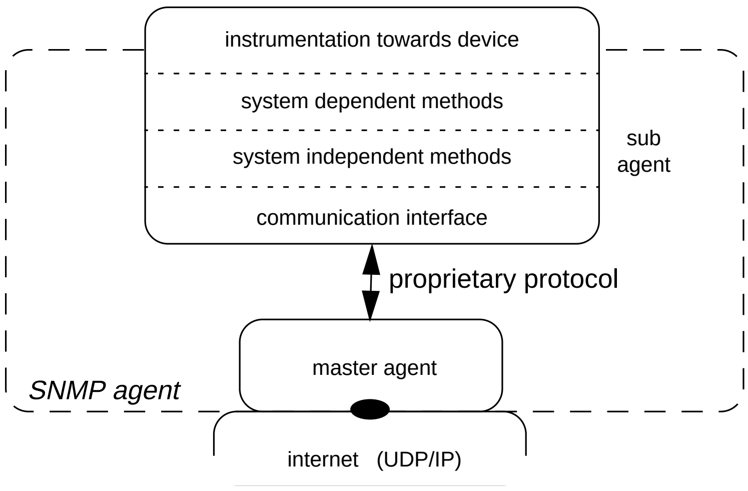


Figure 24: The EMANATE architecture.

Figure 24 shows the general structure of a sub-agent. To communicate with the master agent, the sub-agent includes a special communication interface. This interface is supplied in the form of a library and is used by the system independent methods. These methods can be generated from the MIB specifications, and include a dispatch table. The Application Programmers Interface (API) of the sub-agent consists of the system dependent methods; the communication part is well hidden to the agent implementor. The instrumentation towards the managed device cannot be generated, because it depends on the specific implementation of the device. The main piece of programming that the agent implementor has to do, is to connect the device via the API to the sub-agent.

To develop a sub-agent, the implementor does not need to know the details of the SNMP framework. The SADK allows relatively large MIB to be implemented within a short period of time. It distinguishes five different steps:

- Define the MIB.
  MIBs can be defined from scratch, but in most cases it will be easier to start from an existing definition. MIBs should be defined according to the rules of the Structure of Management Information (SMI) for SNMP version 2.
- Add the instrumentation to the device which has to be managed.
  The instrumentation should provide the required information with which the MIB variables in the MIB can be computed.
- Edit the ‘makefile’.
  This step consists of editing a number of rules and dependencies. These rules define how MIB definitions will be implemented into so called method routines, which are C-code constructs. Next to the method routines, a dispatch table will be generated which defines which variables are available in the MIB definition. The function of the dispatch table is to connect the method routines to the sub-agent’s API.

- Complete the system dependent method routines.
  The SADK tools compile MIB definitions into C-stub routines. These routines form the API, which has to be filled in by the implementor with the appropriate instrumentation. This instrumentation is device specific and can therefore not be generated.
- Compile, link and test the agent.
  A sub-agent can now be created, which forms in conjunction with the master agent the new SNMP agent.

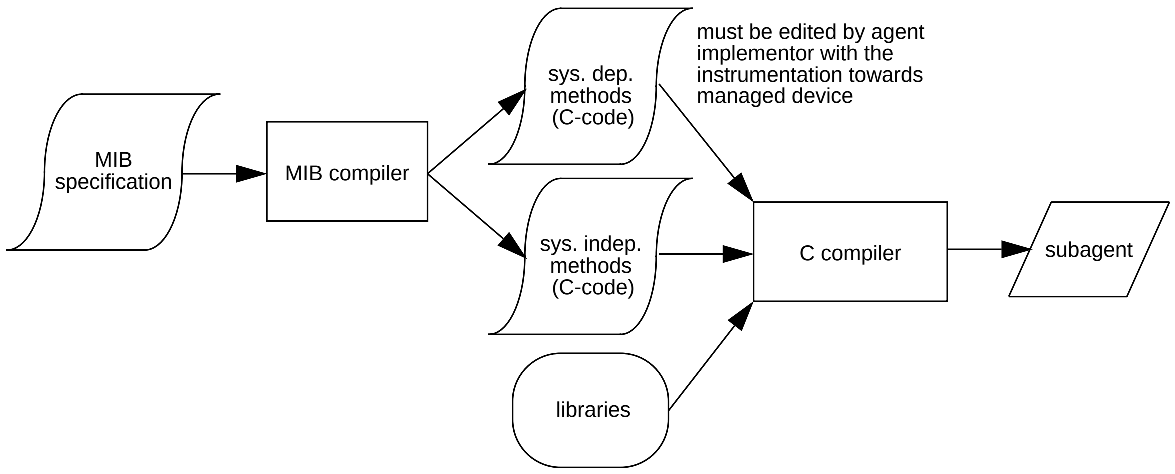

must be edited by agent implementor with the instrumentation towards managed device
sys. dep. methods (C-code)
MIB specification
MIB compiler
sys. indep. methods (C-code)
C compiler
subagent
libraries

Figure 25: Overview of the sub-agent development process.

Figure 25 depicts the development process of the SADK. It shows the way from the MIB specification via the method routines towards the executable sub-agent.

12.2 The method routines.

The method routines can be seen as the pieces of C-code that access and control the managed device. Because parts of this code depends upon the specific way the device is implemented, the implementor has to extend some of these routines manually. To allow easy ports to other implementation environments, each method routine is divided into two parts:
- The System Independent Routines, which check all things that have been defined by the MIB and relate to types, ranges, etc.
- The System Dependent Routines. The implementor extends these routines in order to provide the computational functions of the concrete variables. This involves translating internal types and values into the system independent types, as used by the System Independent Routines.

The computational functions consist of two sorts of processing. These are the GET methods that retrieve values (GET, GET-NEXT and GET-BULK) and the SET methods that change values (SET). In the following two subsections these functions are described in somewhat more detail.

12.2.1 GET processing.

The get processing consists of GET methods which are invoked when the value of a MIB variable has to be computed. Since there are two sorts of MIB variables, scalar variables and tabular variables, there are also two types of GET methods. Based on some examples [43], these methods will be explained below.

12.2.1.1 GET method for scalar variables.

Code 1: MIB definition for two scalar variables

```
examples OBJECT IDENTIFIER ::= { enterprises 785 10 }

integerScalar OBJECT-TYPE
         SYNTAX  INTEGER
         ACCESS  read-only
         STATUS  mandatory
         DESCRIPTION "An example of an integer."
         ::= { examples 1 }

octetStringScalar OBJECT-TYPE
         SYNTAX OCTET STRING (SIZE (0..255))
         ACCESS  read-only
         STATUS  mandatory
         DESCRIPTION "An example of an octet string."
         ::= { examples 2 }
```

After compilation the following stub routine is generated for the complete group. This compilation involves all scalars with the same prefix, i.e. OID ‘examples’, and will combine these variables in a single method.

Code 2: generated stub function for scalars

```
examples_t *
k_examples_get(serialNum, contextInfo, nominator)
   int serialNum;
   ContextInfo *contextInfo;
   int nominator;
{
#ifdef NOT_YET
   static examples_t examplesData;

   /*
    * put your code to retrieve the information here
    */

   examplesData.integerScalar = ;
   examplesData.octetStringScalar = ;
   SET_ALL_VALID(examplesData.valid);
   return(&examplesData);
#else /* NOT_YET */
   return(NULL);
#endif /* NOT_YET */
}
```

The resulting method is a family structure, in the example this is ‘examples_t’. A family structure is a group of scalar (concrete) variables with the same prefix and a variable indicating whether a scalar variable is valid (correct). In the example, the prefix is the OID ‘examples’.

The ‘serialNum’ is used to indicate a unique number which refers to the request PDU. This number could be used to gain performance by using a caching scheme.
The ‘contextInfo’ is used when concrete values depend on the context in which they are defined. For instance, if a variable exists in a certain context only, the method must have the information to check the context.
The ‘nominator’ indicates whether a variable of the family structure is filled with the correct data. This value should correspond to the variable in the family structure indicating whether a scalar variable is valid.

The method shown above has to be edited by the implementor in order to compute the concrete values of the family. In the example, a random number is returned as both type ‘INTEGER’ and type ‘OCTET STRING’.

Each time this method is invoked to compute a concrete variable, a complete family structure is returned. That means that a manager performing a ‘GET(examples.1.0, examples.2.0)’ receives two different values. To avoid this, the ‘serialNum’ variable should be used to cache the random number; each time the method is invoked it should check whether the ‘serialNum’ is changed. (This caching is not shown in the example.)

Code 3: an instrumented method routine.

```
examples_t *
k_examples_get(serialNum, contextInfo, nominator)
   int serialNum;
   ContextInfo *contextInfo;
   int nominator;
{
   static examples_t examplesData;
   unsigned long number;
   char str[30];

   number = random();
   str = ltoa(number);

   examplesData.integerScalar = number;
  examplesData.octetStringScalar = MakeOctetStringFromText(str);
   SET_ALL_VALID(examplesData.valid);
   return(&examplesData);
}
```

12.2.1.2 GET method for tabular variables.

Code 4: MIB definitions for a table containing random numbers.

```
examples   OBJECT IDENTIFIER ::= { enterprises 785 10 }

exampleTable OBJECT-TYPE
         SYNTAX  SEQUENCE OF ExampleEntry
         ACCESS  not-accessible
         STATUS  mandatory
         DESCRIPTION  "A list of a table with random number."
         ::= { examples 3 }

exampleEntry OBJECT-TYPE
         SYNTAX  ExampleEntry
         ACCESS  not-accessible
         STATUS  mandatory
         DESCRIPTION  "A table entry."
              INDEX   { tableIndex }
              ::= { exampleTable 1 }

ExampleEntry ::=  SEQUENCE {
                  tableIndex
                      INTEGER,
                  tableRandomNumber
                      INTEGER
               }
```

```
tableIndex OBJECT-TYPE
         SYNTAX  INTEGER
         ACCESS  not-accessible
         STATUS  mandatory
         DESCRIPTION "The index of the table"
         ::= { exampleEntry 1 }

tableRandomNumber OBJECT-TYPE
         SYNTAX  INTEGER
         ACCESS  read-only
         STATUS  mandatory
         DESCRIPTION "The random number"
         ::= { exampleEntry 2 }
```

After compilation the following stub routine is generated for the complete table. This compilation involves all scalars with the same prefix, i.e. OID ‘exampleEntry’, and combines these in a single method.

Code 5: generated stub routine.

```
exampleEntry_t *
k_exampleEntry_get(serialNum, contextInfo, nominator,
searchType, tableIndex)
   int serialNum;
   ContextInfo *contextInfo;
   int nominator;
   int searchType;
   SR_INT32 tableIndex;
{
#ifdef NOT_YET
   static exampleEntry_t exampleEntryData;

   /*
    * put your code to retrieve the information here
    */

   exampleEntryData.tableIndex = ;
   exampleEntryData.tableRandomNumber = ;
   SET_ALL_VALID(exampleEntryData.valid);
   return(&exampleEntryData);
#else /* NOT_YET */
   return(NULL);
#endif /* NOT_YET */
}
```

The result of the method is a complete family structure, in this example ‘exampleEntry_t’.
The parameters ‘serialNum’, ‘contextInfo’ and ‘nominator’ are equal to those in the methods for scalar variables. The difference is the additional ‘searchType’ and ‘tableIndex’ parameter.
The ‘searchType’ indicates whether the next variable has to be returned or not. The possible values for ‘searchType’ are ‘EXACT’ and ‘NEXT’. The values depend on the GET or GET-NEXT operation received from a manager.
The ‘tableIndex’ indicates the entry on which the action has to be performed. In this example only one index is used, but multiple indexes are possible.

The agent implementor has to edit this method in order to compute the concrete values. In this example, the value of the index number is returned and the number has a random value between 1 and 10.

The example is coded in the following manner:

- The correct entry has to be selected. This is done in two steps:
  - distinguish whether the ‘EXACT’ or ‘NEXT’ value has to be returned,
  - check whether the index is within the range.
- Filling the entry structure with the index and its associated random number.

Code 6: an instrumented stub routine for a table with 10 random numbers

```
exampleEntry_t *
k_exampleEntry_get(serialNum, contextInfo, nominator,
searchType, tableIndex)
   int serialNum;
   ContextInfo *contextInfo;
   int nominator;
   int searchType;
   SR_INT32 tableIndex;
{
   static exampleEntry_t exampleEntryData;
   int index;

   /* selection of correct entry */
   if ( searchType == EXACT ) {
      if (( tableIndex <= 1 ) && ( 10 <= tableINdex ))
      /* in range, thus value is tableIndex */
         index = tableIndex;
      else
      /* out of range, thus NoSuchInstance */
         return(NULL);
   else /* searchType == NEXT */
      if (( tableIndex <= 0 ) && ( 9 <= tableINdex ))
      /* in NEXT range, thus value is NEXT tableIndex */
        index = tableIndex + 1;
     else
     /* out of range, thus noSuchInstance */
        return(NULL);

   /* filling of entry structure */
   exampleEntryData.tableIndex = index;
   exampleEntryData.tableRandomNumber = random();
   SET_ALL_VALID(exampleEntryData.valid);
   return(&exampleEntryData);
}
```

12.2.2 SET processing

In order to handle the ‘as-if-simultaneous’ requirements of SNMP sets, 4 types of methods are necessary for scalar and tabular variables.

Code 7: MIB definition for a writeable scalar variable

```
examples   OBJECT IDENTIFIER ::= { enterprises 785 10 }

writeableScalar OBJECT-TYPE
         SYNTAX  INTEGER
         ACCESS  read-write
         STATUS  mandatory
         DESCRIPTION "An example of a writable variable."
         ::= { examples 4 }
```

After compilation the following stub routines for the set-processing are generated (at group level).

Code 8: generated stub function for writeable scalars

```
#ifdef SETS
int
k_examples_test(object, value, dp, contextInfo)
    ObjectInfo     *object;
    ObjectSyntax   *value;
    doList_t       *dp;
    ContextInfo    *contextInfo;
{
    return NO_ERROR;
}

int
k_examples_ready(object, value, doHead, dp)
    ObjectInfo     *object;
    ObjectSyntax   *value;
    doList_t       *doHead;
    doList_t       *dp;
{
    dp->state = ADD_MODIFY;
    return NO_ERROR;
}

int
k_examples_set(data, contextInfo, function)
   examples_t *data;
   ContextInfo *contextInfo;
   int function;
{
   return GEN_ERROR;
}
#endif /* SETS */

#ifdef SR_SNMPv2
#ifdef SR_examples_UNDO
/* add #define SR_examples_UNDO in sitedefs.h to
 * include the undo routine for the examples family.
 */
int
examples_undo(doHead, doCur, contextInfo)
    doList_t       *doHead;
    doList_t       *doCur;
    ContextInfo    *contextInfo;
{
   return UNDO_FAILED_ERROR;
}
#endif /* SR_examples_UNDO */
#endif /* SR_SNMPv2 */
```

The generated set-processing stubs consists of 4 method, of which the last one is only for SNMP version 2.

- ‘k_examples_test’ method checks whether the implementation allows to perform this set. This test is not dependent on SYNTAX constraints defined in a MIB specification, because those kind of checks are done by the system independent method routines. This method is for local (system dependant) testing and does not depend on other values in the set request or current values of other variables. For instance, this function checks the range of an implementation dependent ‘INTEGER’.
- ‘k_examples_ready’ method is used for global testing. That is, it checks the consistency of the system with respect to other MIB variables.
- ‘k_examples_set’ is responsible for adding, modifying, or deleting of the values. This method is only invoked if all ‘test’ and ‘ready’ methods have been succeeded.
- ‘examples_undo’ method is only available in agents supporting SNMPv2.

The agent implementor has to edit these methods in order to provide the correct instrumentation for the set-processing. In this example, a global variable is set.

Code 9: instrumented method routine.

```
#ifdef SETS
int
k_examples_test(object, value, dp, contextInfo)
    ObjectInfo     *object;
    ObjectSyntax   *value;
    doList_t       *dp;
    ContextInfo    *contextInfo;
{
    return NO_ERROR;
}

int
k_examples_ready(object, value, doHead, dp)
    ObjectInfo     *object;
    ObjectSyntax   *value;
    doList_t       *doHead;
    doList_t       *dp;
{
    dp->state = ADD_MODIFY;
    return NO_ERROR;
}

int
k_examples_set(data, contextInfo, function)
   examples_t *data;
   ContextInfo *contextInfo;
   int function;
{
   if ( VALID( I_writeableScalar, data->valid ) ) {
      globalVariable = data->writeableScalar;
      return NO_ERROR;
   }
   return GEN_ERROR;
}
#endif /* SETS */

#ifdef SR_SNMPv2
#ifdef SR_examples_UNDO
/* add #define SR_examples_UNDO in sitedefs.h to
 * include the undo routine for the scalar family.
 */
int
examples_undo(doHead, doCur, contextInfo)
    doList_t       *doHead;
    doList_t       *doCur;
    ContextInfo    *contextInfo;
{
   return UNDO_FAILED_ERROR;
}
#endif /* SR_examples_UNDO */
#endif /* SR_SNMPv2 */
```

If set-processing is required for table, then the family level is replaced by the entry level which is in principle equivalent to the GET/GET-NEXT processing.

13 Sub-agent design

Figure 26 shows a WWW server managed by an extensible agent. Besides the WWW MIB, the agent also implements the MIB-II. Because this MIB-II implementation has been purchased from SNMP Research, the remainder of this section focuses on the WWW sub-agent.

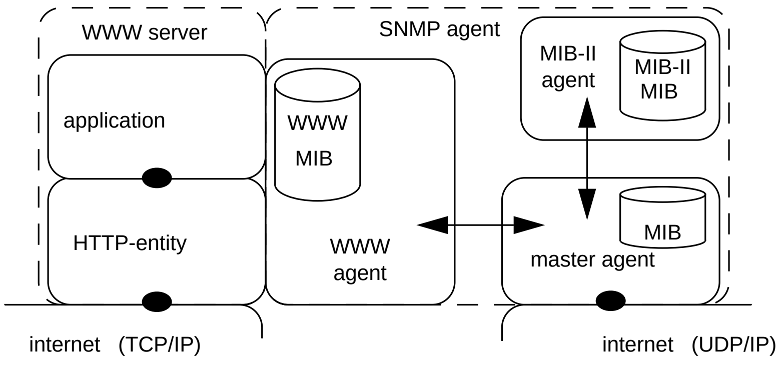


Figure 26: Management structure

The WWW sub-agent is designed according to the rules explained in Section 12, i.e.:
- the WWW server has to be extended with management instrumentation,
- the WWW MIBs have to be compiled to generate the stub-routines (create a sub-agent without the instrumentation of the WWW server),
- to compile all sources, a ‘makefile’ has to be edited,
- the stub-routines have to be instrumented to compute the concrete values (connect the stub-routines to the management instrumentation), and
- the WWW sub-agent has to be tested.

To obtain management information from the WWW server, the following approaches are possible:
- The direct access approach. In this approach WWW server code has to be extended/modified in order to adopt the MIB population. This approach was rejected by the project, because it implied that for each server version the extensions/modifications must be made again. If on the long term implementors of WWW servers include MIB support, this approach may become the preferred one.
- The logfile scanning approach. The advantage of this approach is that WWW server software need not be altered to include management instrumentation. Another important advantage is that the same sub-agent implementation can be used with different WWW servers, because the logfile format is defined according to a de facto standard. For these reasons the logfile scanning approach was selected by our project. A disadvantage of scanning logfiles is that certain variables that are not included in the logfile, cannot be computed. The second major disadvantage is that the management instrumentation will be limited to monitoring; changing the behaviour of the WWW server by modification of MIB variables will not be possible.

To increase performance, it was decided to implement all WWW MIBs (IS, RS and HTTP) within a single sub-agent.

13.1 Management instrumentation

Figure 27 shows the selected approach. The logfile is filled by the WWW application and the HTTP entity. The logfile is read by the WWW sub-agent, and used to compute / update the

WWW MIB variables. These variables are read and set (remember the manager could update certain tables by setting certain variables) by the master agent.

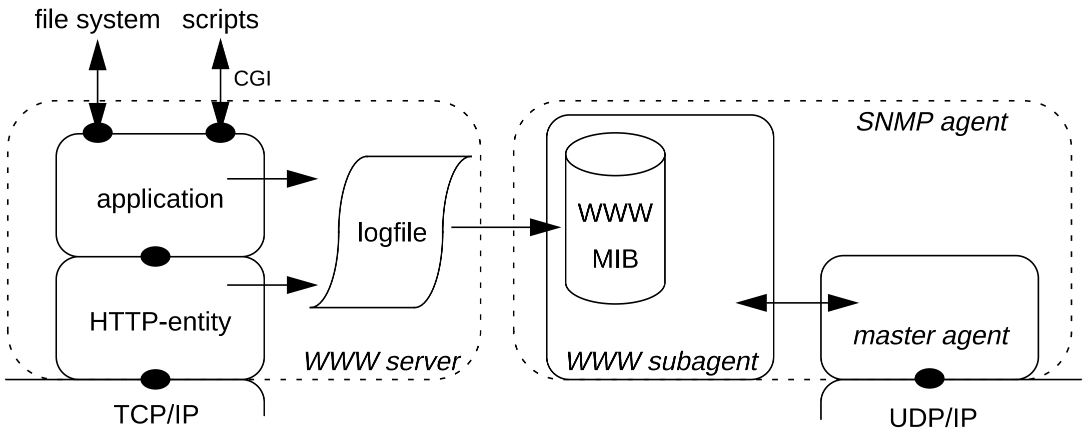


Figure 27: The logfile approach

13.2 The time/space problem.

The size of logfiles that are generated by busy WWW servers can easily grow to many Megabytes. The impact of interpreting such logfiles, with respect to processing time as well as memory usage, should not be underestimated. In fact we are confronted with the traditional time/space problem which, as always, has two extreme solutions:

- Compute the value of a MIB variable only after a specific request is obtained from the manager. This solution demands that after each SNMP request a scan is made through the entire logfile.
- Compute / update MIB variables each time a new entry is detected at the end of the logfile. This solution demands that the sub-agent continuously monitors the logfile, even if no SNMP requests are made.

Although both solutions provide the desired functionality, there are important differences. The first solution saves memory and guarantees that values are always up-to-date (because they are computed at the time the SNMP request is made). The major disadvantage of this solution is the time needed to compute the requested values. In case the size of the logfile has become many megabytes, scanning the file may consume so much time that the manager gets a time-out.

With the second solution pre-computed MIB values are stored in memory. On receipt of an SNMP request, the requested value can immediately be returned to the manager. The main disadvantage of this approach is that a large amount of memory may be needed to cache the MIB variables. Also the computing time that is needed to constantly monitor the logfiles and update the MIB variables (even if no SNMP requests are made) may become a problem. The usual way to cope with this problem, is to monitor the logfiles and update the MIB variable periodically instead of constantly. This introduces another disadvantage however: MIB values may not always be up-to-date.

The solution selected for the project is a hybrid approach. For most variables, the second solution has been implemented: the logfiles are constantly monitored and many variables are kept in main memory. For two tables, the most frequent user table and the document table, the first solution has been selected. This is because storing these tables requires a lot of memory. The WWW MIBs have therefore been defined in such a way that these two tables could be computed upon request of the manager only. To avoid superfluous updates, a timestamp variable, denoting the time the table was last updated, has been associated with each table.

13.3 Functional decomposition.

To structure the implementation, a functional decomposition was made. This decomposition divides the agent into functional elements with well defined behaviour. As illustrated in Figure 28, the main function of the agent is to translate the logfile information into concrete values for the MIB variables.

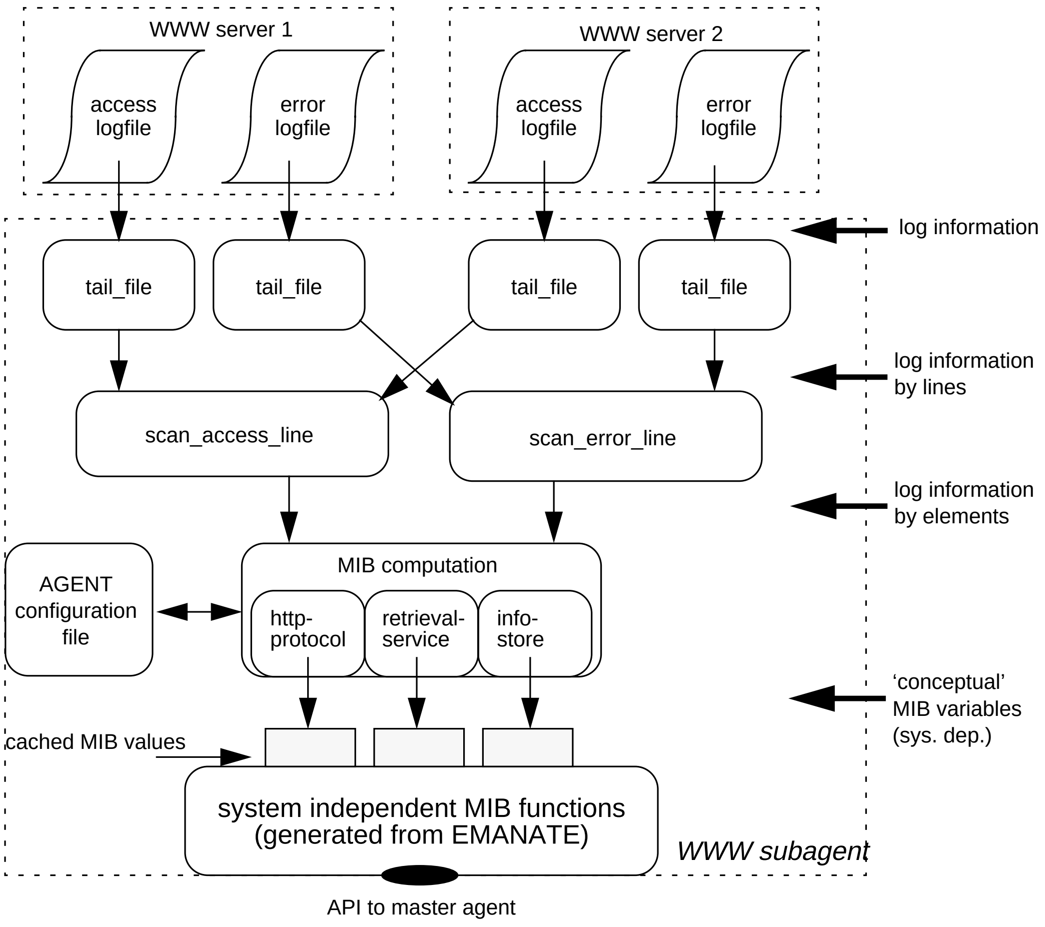


Figure 28: Functional decomposition of a sub-agent

The figure shows how the access and error logfiles of possibly multiple WWW servers[1] are scanned (tailed) by the *tail_file* functions. The result of these functions are processed by the *scan_access_line* and *scan_error_line* functions. These functions create from individual logfile lines meaningful elements of informations. On its turn, the *MIB_computation* function updates all cached MIB variables.

1. Remember that multiple servers may run on a single system.

14 Sub-agent details

This section describes the functional elements identified in Figure 28. It provides a detailed view of the instrumentation and the way the information from the logfiles is processed in order to populate (compute) the MIB variables.

14.1 Tail_file function

After each server access a line containing access or error information is appended to the logfile. The agent keeps track of these lines and computes from these the MIB variables values. The function that keeps track of new lines is called ‘Tail_file’. This function shows the following behaviour:

- If a correct line is read from the logfile, the function returns this line (success).
- If the end-of-file is reached, the function returns an empty line (success).
- If an incorrect line is read from the logfile, the function skips this line and proceeds reading (error-recovery).

14.2 Scan_*_line function

The lines returned by the ‘Tail_file’ function are character strings. In order to process and compute MIB variables from these lines, they have to be split into basic line elements. This conversion is executed by the *scan_*_line* functions. For the access logfile the basic elements are defined by the Common Logfile Format. Unfortunately the format for the error logfile has not been standardized. Two scan functions have therefore been implemented: one for the access logfile (this function can be used with every type of server) and one for the error logfile (this function must be modified in case different types of servers will be managed).

The function for the access logfile is called *scan_access_line* and returns the following:

- success: a correctly filled structure of type ‘AccessLine’ is returned,
- error: no structure is returned.

The function for the error logfile is called *scan_error_line* and returns the following:

- success: a correctly filled structure of type ‘ErrorLine’ is returned,
- error: no structure is returned.

14.3 MIB computation

MIB computation, which is needed to compute the concrete MIB values, has been divided into three different parts:

- HTTP MIB computation, see Subsection 14.3.1.
- RS MIB computation, see Subsection 14.3.2.
- IS MIB computation, see Subsection 14.3.3.

To compute concrete MIB values, the following sources have been used:

- The access logfile, which records all accesses executed at the WWW server.
- The error logfile, which records all server (access) errors.
- System internal functions, like OS functions to compute the time and day.
- A special agent configuration file, which has been specifically introduced for the purpose of WWW server management. It contains, in text format, information like the applications that create dynamic information via the CGI interface.

Since current WWW server software does not yet include management instrumentation, the direct access method (see Section 13) cannot not be used and certain fields cannot be computed. Also some variables are only meaningful in clients/proxies; these variables are not computed, but return a zero value.

14.3.1 HyperText Transfer Protocol MIB computation

System group

The system group contains a table with entity specific information. Most of this information is read from a static agent configuration file. This configuration file contains concrete values for some MIB variables, but also pointers to places where additional information can be found. Figure 29 shows how the system table has been populated.

variable	population
httpEntityIndex	system internal
httpEntityProtocol	agent configuration file
httpEntityDescription	agent configuration file
httpEntityContact	http server configuration file
httpEntityProcotolVersion	agent configuration file
httpEntityVendor	agent configuration file
httpEntityVersion	agent configuration file
httpEntityObjectID	agent configuration file
httpEntityAddress	http server configuration file
httpEntityPort	http server configuration file
httpEntityIpAddress	system internal (DNS)
httpEntityLastInitialisation	agent configuration file
httpEntityType	agent configuration file

Figure 29: System table

Statistics group

The statistics group contains three tables: the summary table, the request table and the response table. Figure 30, Figure 31 and Figure 32 show how these tables have been computed.

Parts of the summary table’s information can also be found in the request or response table. There are therefore two alternatives to calculate summary table values:
- Direct calculation: the values are derived from the logfiles.
- Indirect calculation: the values are derived from the request or response table.

variable	population
httpSummaryIndex	system internal
httpSummaryRequests	entry from access logfile

Figure 30: Summary table

variable	population
httpSummaryRequestsErrors	- (direct calculation)
httpSummaryRequestsDiscards	- (direct calculation)
httpSummaryResponses	entry from access logfile
httpSummaryResponsesErrors	- (direct calculation)
httpSummaryResponsesDiscards	- (direct calculation)
httpSummaryinUnknowns	- (direct calculation)
httpSummaryInBytes	- (direct calculation)
httpSummaryOutBytes	bytes from access logfile entry
httpSummaryTimeOuts	time out from error logfile entry

Figure 30: Summary table

The request table, which contains entries for each type of request, is indexed by both the entity which is monitored as well as the request type. Although the HTTP protocol uses names to identify the various request types, the MIB uses an enumeration of INTEGERS. The mapping from INTEGERS to names is defined by a textual convention.

variable	population
httpRequestIndex	system internal
httpRequestMethodIndex	URI from access logfile entry / protocol standard
httpRequestInCount	entry from access logfile
httpRequestInLastTime	date and time from access logfile entry
httpRequestOutCount	-
httpRequestOutLastTime	-

Figure 31: Request table

The response table specifies all individual response types.

variable	population
httpResponseIndex	system internal
httpResponseMethodIndex	system internal / protocol standard
httpResponseInCount	-
httpResponseInLastTime	-
httpRequestOutCount	entry from access logfile
httpRequestOutLastTime	date and time from entry of access logfile

Figure 32: Response table

TimeOuts group

This group of the HTTP protocol counts the number of time-outs and shows the address of the remote system that causes the time-out. Figure 33 shows how the variables have been computed.

variable	population
httpTimeOutTableSize	system internal
httpTimeOutTable - httpTimeOutEntityIndex	system internal
httpTimeOutNumber	system internal
httpTimeOutRemoteAddress	remotehost from error logfile entry
httpTimeOutTime	remotehost from error logfile entry

Figure 33: TimeOut group

The length of the TimeOut table can be determined by the manager by setting the 'httpTimeOutTableSize' variable. The maximum value of this variable is limited by the pre-defined 'maxhttpTimeOutTableSize'. The table is implemented by means of a circular buffer.

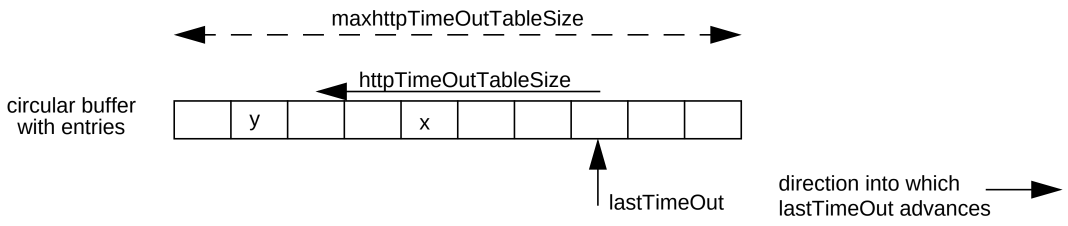

maxhttpTimeOutTableSize
httpTimeOutTableSize
circular buffer with entries
y
x
lastTimeOut
direction into which lastTimeOut advances

Figure 34: Implementation of the TimeOut table

Figure 34 shows this implementation. The 'lastTimeOut' variable points to the most recent time-out entry and advances to the right if new time-outs occur. If a client requests information from the entry that has index *n*, the circular buffer should be searched *n* places into the opposite direction (of course *n* should not exceed 'httpTimeOutTableSize').

> ***Example:*** If the fourth table entry (index = 4) is requested, the contents of entry *x* is returned. In case the seventh table entry is requested, no value is returned since *y* is outside the 'httpTimeOutTableSize'. In such case the SNMP response carries as error indication 'noSuchInstance'.

14.3.2 Retrieval Service MIB computation

Statistics group

This group counts the number of Retrieval Service primitives at the WWW server side.

variable	population
rsTotalRequests	-
rsTotalIndications	entry from access logfile
rsTotalResponses	entry from access logfile
rsTotalConfirmations	-

Figure 35: Statistics group

Quality of Service group

Most Quality of Service values cannot be computed from the logfiles. Therefore only the counters ‘rsNumberOfErrors’ and ‘rsNumberOfTimeOuts’ have been implemented.

variable	population
rsDelayTable - rsSource rsDestination rsDelay	remote host from access logfile entry / system remote host from access logfile entry / system -
rsNumberOfErrors	entry from error logfile
rsnumberOfTimeOuts	entry from error logfile
rsThroughputTable - rsClient rsThroughput	remote host from access logfile entry -

Figure 36: Quality of Service group

14.3.3 Information Store MIB computation

General group

As shown in Figure 37, the general group is primarily populated from the agent configuration file.

variable	population
isName	agent configuration file
isOrganisation	agent configuration file
isContact	agent configuration file
isLastInitialisation	system internal
isSupportedMediaTypes	agent configuration file
isApplDependancyTable - applIndex applResourceIdentifier applProcessName applVersion applUptime applOperStatus applLastChange applLastActivity applFailedActivities	system internal agent configuration file agent configuration file - - ‘proces status’ - command - - -
TopicTable - topic	agent configuration file

Figure 37: General group

Population of the application dependency table is not straightforward, since the ‘associated processes’ can not automatically be determined. It was therefore decided to introduce an agent configuration file, which contains the names of all associated applications. After the application’s name has been read from the file, the status of the application process is computed by using the ‘process status’ command. It should be noted that the system administrator is responsible for the initialization and maintenance of the table.

Also the Topic table is populated from the agent’s configuration file. The administrator has to enter the topics in lexical graphical order, because the agent cannot order them itself.

Access group

Figure 38 shows how the access group is populated.

variable	population
isNumberOfAccess	entry from access logfile
isNumberOfBytesIn	-
isNumberOfBytesOut	bytes from access logfile entry
isDomainTable - domainName domainAccesses	remotehost from access logfile entry remotehost from access logfile entry
isAccessOfLastNDaysTableSize	system internal
isAccessOfLastNDaysTable - daysIndex daysAccesses	date and time from access logfile entry date and time from access logfile entry
isMostFrequentUserTableSize	system internal
isMostFrequentUserTableRefresh	system internal
isMostFrequentUserTabledate	system internal
isMostFrequentUserTable - mfIndex mfUserName mfUserAccesses	system internal remotehost from access logfile entry remotehost from access logfile entry
isMostRecentUserTableSize	system internal
isMostRecentUserTable - mrIndex mrUserName mfTime mfDocument	system internal remotehost from access logfile entry date and time from access logfile entry req URL from access logfile entry

Figure 38: Access group

Error group

The MIB definition does not prescribe which specific errors have to be counted, but leave that decision up to the agent implementor. In our proof of concept study, it was decided to record the following errors:
- CGI errors (script errors).
- Document errors (access failures).
- Transfer aborted messages.

variable	population
errorIndex	system internal
errorDescription	system internal
errorCount	sort of error from error logfile entry
errorLastTime	date and time from error logfile entry

Figure 39: Error Table

Document group

Figure 40 shows how the document group is populated.

variable		population
	isDocumentTableSize	system internal
	isDocumentTableRefresh	system internal
	isDocumentTableDate	system internal
isDocumentTable -	documentIndex	system internal
	documentName	remotehost of entry from access logfile
	documentAccesses	remotehost of entry from access logfile
	documentAccessRights	system internal
	documentSize	system internal
	documentErrors	-
	documentUpdate	-
	documentAccessesAtLastUpdate	-
	documentLastAccesses	system internal
	documentType	-

Figure 40: Document group

Part V

Conclusions and Recommendations

15 Conclusions and recommendations

The main goal of the project, to demonstrate that information retrieval applications can be managed, was achieved. The project selected the World Wide Web (WWW) to implement the information retrieval applications, and SNMP to manage these applications. After the decision was made to use SNMP, an alternative management technique, called Web Based Management (WBM) appeared. It is recommended that subsequent projects seriously investigate the merits of WBM.

15.1 MIB definition

The idea of dividing the WWW MIB into three separate MIBs (IS, RS and HTTP MIB) is a good idea. It allows to separate concerns and it makes future modifications easier. If, for example, the decision is made to use another information transport protocol, the HTTP MIB needs to be replaced only.
Another advantage of structuring the management information into independent MIBs, became apparent at the time the project decided to collaborate with others, who were also interested in WWW management, but wanted to focus on the definition of HTTP related management information. The fact that we had separate MIBs, allowed us to work together on the definition of the HTTP MIB but work individually on the definition of the two other MIBs. The collaboration took place via the HTTP MIB mailinglist.

At the end of the project the three MIBs were presented as ‘internet-drafts’ to the IETF. The exact text of these drafts is included in the appendixes. The HTTP MIB is now being further progressed by the members of the HTTP MIB mailinglist.

The IS MIB contains an application dependency table (see Subsection 10.1), which keeps track of the applications that provide the server with dynamic WWW information. Such an application may for instance be a database that communicates to the WWW server via the CGI interface. At the time the application dependency table was defined, an IETF working group just started to define a special Application MIB. Now that some results of this IETF group are available, it is suggested to investigate the possibility to remove the application dependency table from the IS MIB and use the Application MIB instead.
Other MIB aspects that may need further discussion, are the access rights that are defined in the document group of the IS MIB (see Subsection 10.4) and the TimeOut table of the HTTP MIB (see Subsection 8.3).

The amount of data needed to store all possible pieces of WWW management information may not be neglected. Also the processing time to calculate all MIB values should not be underestimated. To control memory usage and CPU demand, it was decided to introduce special MIB variables that allow managers to specify when certain tables should be updated and what sizes various tables should have (see for instance Subsection 10.2).
At the end of the project an idea appeared on the mailinglist to reduce the number of SNMP transactions that are needed to update tables at the manager side, by using a time filter such as described by the RMON-2 MIB.

15.2 Agent implementation

Most parts of the MIBs could be implemented within the time frame of the project. An exception is the RS MIB, which could only be partially implemented.

The implementation is based on the EMANATE sub-agent development kit, which is produced by SNMP Research. Since our software is implemented as a sub-agent, it cannot be used without a master agent. Master agents can not be distributed freely, but should be purchased from SNMP Research.

To allow free distribution of WWW management software, the UT also started to develop a Public Domain version of the agent. This version, which is produced outside the scope of this project, uses the Scotty package and will be based on the latest version of the HTTP MIB, as defined by the mailinglist.

WWW servers do not yet include management APIs. The project therefore decided to populate the MIBs from the logfiles that are maintained by every server. The use of logfiles has some disadvantages however; logfiles can for instance not be used to modify the server's behaviour. Although this didn't turn out to be a problem within the context of our project, manufacturers of WWW servers are still advised to include management APIs.

The format of access logfiles is defined by a de facto standard, which implies that our software can be used with a variety of servers. Unfortunately the error logfiles have not been standardized, which means that our software will not be able to provide all available error information with each kind of server. The servers that are supported by our software, are those from NCSA and APACHE.

16 References

[1] Autrata M., Strutt C.: “DME Framework and Design”, in: Network and Distributed Systems Management, Chapter 23, Addison-Wesley Publishing Company, 1994

[2] CCITT: “Recommendation M.3010, Principles for a Telecommunications Management Network”, Geneva 1992

[3] CMIP Run!; “New APIs for Management Data”, Vol 3, No 2, 2nd Q ’94, page 11

[4] Comer D.E. and Stevens D.L.: “Internetworking with TCP-IP Vol.II - Design, Implementation & Internals”, Englewood Cliffs NJ, Prentice Hall, 1991, ISBN 0-13-134677-6.

[5] Embry J., Manson P., Milham D., “Interoperable Network Management: OSI/NM Forum Architecture and Concepts”, in: Proceedings of the IFIP TC6/WG 6.6 Second International Symposium on Integrated Network Management, page 29-44, North-Holland, 1991

[6] Feih S.: “SNMP - A guide to network management”, McGraw-Hill, Inc., 1995, ISBN

[7] ISO 7498-4: “Information Processing Systems - Open Systems Interconnection - Basic Reference Model - Part 4: Management Framework”, Geneva, 1989

[8] ISO 9596: “Information Processing Systems - Open Systems Interconnection - Common Management Information Protocol”, Geneva, 1991

[9] Murrill B.: “OMNIPoint: An Implementation Guide to Integrated Networked Information Systems Management”, in: Proceedings of the IFIP TC6/WG 6.6 Third International Symposium on Integrated Network Management, page 405-418, North-Holland, 1993

[10] Network Management Forum: “Discovering Omnipoint - a common approach to the Integrated Management of Networked Information Systems”, 1993

[11] Perkins D. T.: “Understanding SNMP MIBs, revision 1.1.5”, July 1992

[12] Perkins, D.: “User’s guide for SMICng - The SNMP MIB Information Compiler”, SynOptics Communications, Inc., U.S.A., November 1994.

[13] Pras A.: “Network Management Architectures”, Ph.D. thesis University of Twente, the Netherlands, CTIT 95-02, 1995

[14] RFC 1028: “Simple Gateway Monitoring Protocol”, Davin J., Case J.D., Fedor M., Schoffstall M.L., November 1987

[15] RFC 1052: “IAB recommendations for the development of Internet network management standards”, Cerf V.G., April 1988

[16] RFC 1155, “Structure and identification of Management Information for TCP/IP-based internets”, Rose M.T., McCloghrie K., May 1990

[17] RFC 1157: “Simple Network Management Protocol (SNMP)”, Case J.D., Fedor M., Schoffstall M.L., Davin C., May 1990

[18] RFC 1189: “Common Management Information Services and Protocols for the Internet (CMOT and CMIP)”, Warrier U.S., Besaw L., LaBarre L., Handspicker B.D., October 1990

[19] RFC 1213: “Management Information Base for network management of TCP/IP-based internets: MIB-II”, McCloghrie K., Rose M.T., March 1991

[20] RFC 1441: “Introduction to version 2 of the Internet-standard Network Management Framework”, Case J., McCloghrie K., Rose M.T., Waldbusser S., April 1993

[21] RFC 1442: “Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)”, Case J., McCloghrie K., Rose M., Waldbuster S., April 1993

[22] RFC 1443: “Textual Conventions for version 2 of the Simple Network Management Protocol (SNMPv2)”, Case J., McCloghrie K., Rose M., Waldbuster S., April 1993

[23] RFC 1444: “Conformance Statements for version 2 of the Simple Network Management Protocol (SNMPv2)”, Case J., McCloghrie K., Rose M., Waldbuster S., April 1993

[24] RFC 1445: “Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2)”, McCloghrie K., Galvin J., April 1993

[25] RFC 1446: “Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2)”, McCloghrie K., Galvin J., April 1993

[26] RFC 1447: “Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2)”, McCloghrie K., Galvin J., April 1993

[27] RFC 1448: “Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)”, Case J., McCloghrie K., Rose M., Waldbuster S., April 1993

[28] RFC 1449: “Transport mappings for version 2 of the Simple Network Management Protocol (SNMPv2)”, Case J., McCloghrie K., Rose M., Waldbuster S., April 1993

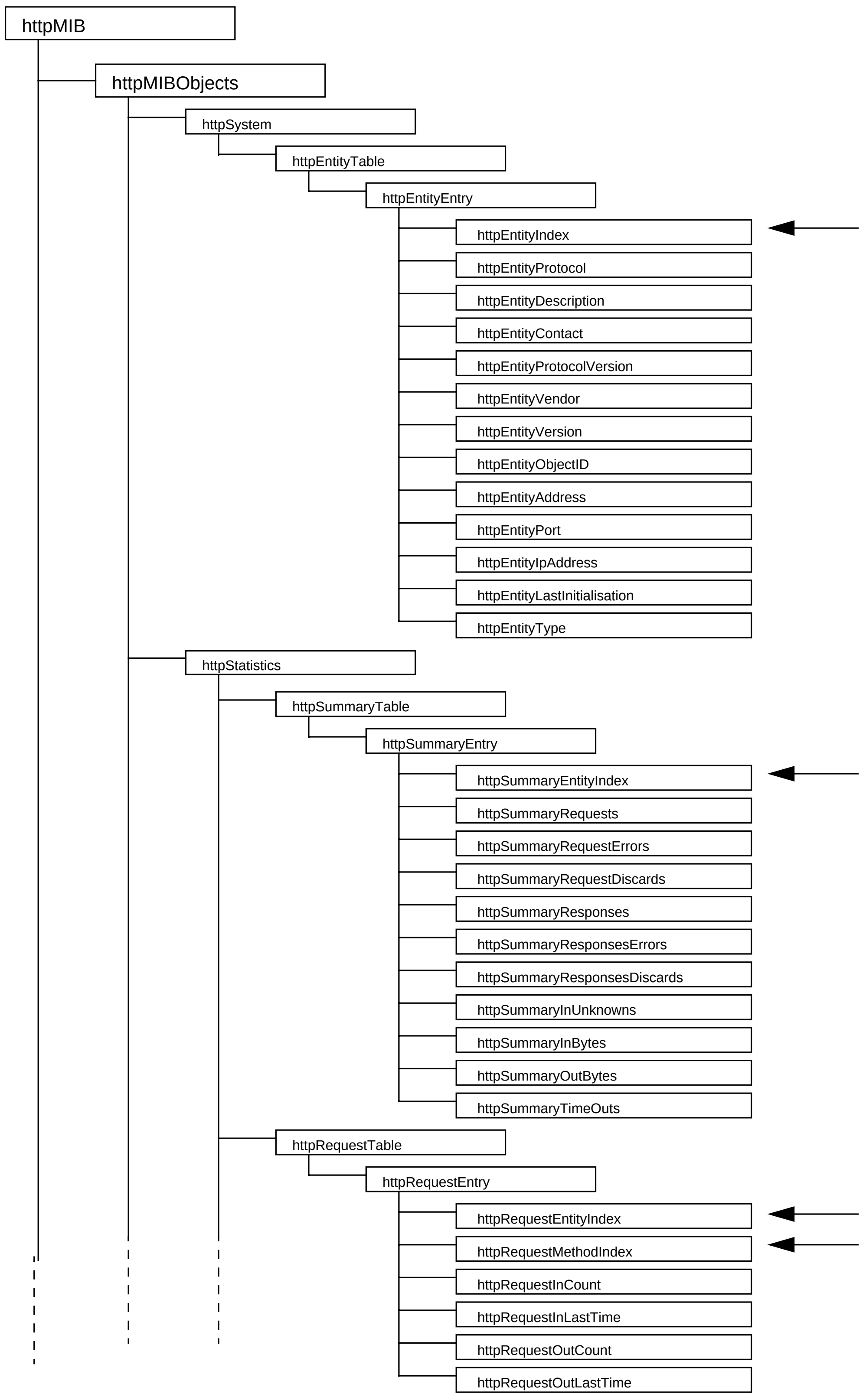
[29] RFC 1450: “Management Information Base for version 2 of the Simple Network Management Protocol (SNMPv2)”, Case J., McCloghrie K., Rose M.T., Waldbusser S., April 1993

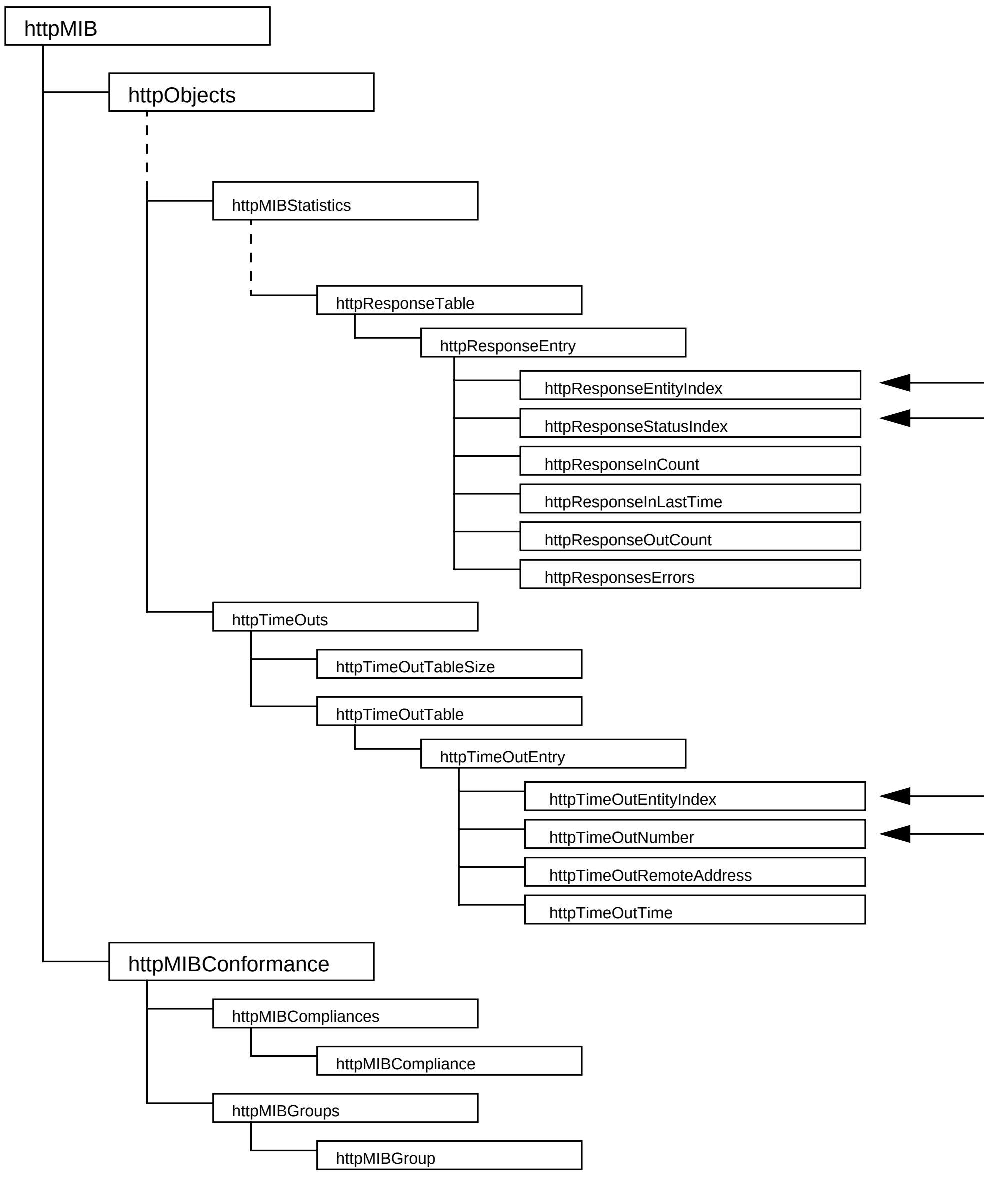
[30] RFC 1451: “Manager-to-Manager Management Information Base”, Case J., McCloghrie K., Rose M.T., Waldbusser S., April 1993

[31] RFC 1452: “Coexistence between version 1 and version 2 of the Internet-standard Network Management Framework”, Case J., McCloghrie K., Rose M., Waldbuster S., April 1993

[32] RFC 1901: “Introduction to Community-based SNMPv2”
[33] RFC 1902: “Structure of Management Information for Version 2 of the Simple Network ManagementProtocol”, Case J., McCloghrie K., Rose M.T., Waldbusser S., January 1996
[34] RFC 1903: “Textual Conventions for Version 2 of the Simple Network Management Protocol”, Case J., McCloghrie K., Rose M.T., Waldbusser S., January 1996
[35] RFC 1904: “Conformance Statements for Version 2 of the Simple Network Management Protocol”, Case J., McCloghrie K., Rose M.T., Waldbusser S., January 1996
[36] RFC 1905: “Protocol Operations for Version 2 of the Simple Network Management Protocol”, Case J., McCloghrie K., Rose M.T., Waldbusser S., January 1996
[37] RFC 1906: “Transport Mappings for Version 2 of the Simple Network Management Protocol”, Case J., McCloghrie K., Rose M.T., Waldbusser S., January 1996
[38] RFC 1907: “Management Information Base for Version 2 of the Simple Network Management Protocol”, Case J., McCloghrie K., Rose M.T., Waldbusser S., January 1996
[39] RFC 1908: “Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework”, Case J., McCloghrie K., Rose M.T., Waldbusser S., January 1996
[40] Rose M.T.: “The Simple Book - Second Edition”, Prentice-Hall International Editions, 1994
[41] Rose M.T., McCloghrie K.: “How to manage your network using SNMP”, 1994
[42] Scott Marcus J.: “Icaros, Alice and the OSF DME”, in: Proceedings of the IFIP TC6/WG 6.6 Fourth International Symposium on Integrated Network Management, page 83-92, Chapman & Hall, 1995
[43] SNMP Research International: “SNMP Research EMANATE Subagent Development Kit - Developer Documentation”, SNMP Research International, Inc., Knoxvile Tennessee U.S.A., December 1994.
[44] Stallings W.: “SNMP, SNMPv2 and CMIP - The Practical Guide to Network Management Standards”, Addison Wesley, 1994

Appendix A: HyperText Transfer Protocol MIB definition




httpMIB
httpObjects
httpMIBStatistics
httpResponseTable
httpResponseEntry
httpResponseEntityIndex
httpResponseStatusIndex
httpResponseInCount
httpResponseInLastTime
httpResponseOutCount
httpResponsesErrors
httpTimeOuts
httpTimeOutTableSize
httpTimeOutTable
httpTimeOutEntry
httpTimeOutEntityIndex
httpTimeOutNumber
httpTimeOutRemoteAddress
httpTimeOutTime
httpMIBConformance
httpMIBCompliances
httpMIBCompliance
httpMIBGroups
httpMIBGroup

Definitions of Managed Objects for HTTP

April 22, 1996

<draft-hazewinkel-httpmib-00.txt>

Harrie Hazewinkel
University of Twente
hazewink@cs.utwente.nl

Eric van Hengstum
University of Twente
hengstum@cs.utwente.nl

Aiko Pras
University of Twente
pras@cs.utwente.nl

Status of this Memo

1. Abstract

This draft defines a MIB to manage the HTTP protocol. The HTTP protocol is used by World Wide Web applications to transfer information between WWW clients and WWW servers. HTTP is based on the request/response paradigm.
The HTTP MIB is especially useful to manage the WWW server-side. The MIB is defined in such a way that it allows multiple servers to run within a single system.

2. TheSNMPv2 Network Management Framework

The SNMPv2 Network Management Frameworkconsists of four major components. They are:

- o STD 17, RFC 1213 [2] defines MIB-II, the core set of managed objects for the Internet suite of protocols.
- o RFC 1901 Introduction to Community-based SNMPv2
- o RFC 1902 Structureof Management Information for Version 2of the SimpleNetworkManagement Protocol (SNMPv2)
- o RFC 1903 Textual Conventions for Version 2of the Simple Network ManagementProtocol (SNMPv2)
- o RFC 1904 Conformance Statements for Version 2 of the Simple Network ManagementProtocol (SNMPv2)
- o RFC 1907 Management Information Base for Version 2of the Simple Network Management Protocol(SNMPv2)
- o RFC 1908 Coexistence between Version 1 andVersion2 of the Internet-standard Network Management Framework

The Framework permits new objects to bedefinedfor thepurposeof experimentationand evaluation.

2.1. Object Definitions

Managedobjects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the subset ofAbstract SyntaxNotation One (ASN.1)defined in the SMI[1]. In particular, each object type is named by an OBJECT IDENTIFIER, an administratively assigned name. The object typetogether with an objectinstance servesto uniquely identify a specific instantiation of the object. For human convenience, we often usea textual string, termed the object descriptor, to refer to the object type.

3. Introduction.

The HTTP protocol instantiates the information transport in the WWW application which is done with protocol entities exchanging information with each other via the underlying service.

The work performed for this MIB is a result of a project performed for the Centre of Earth Observations. The document should not be considered to be complete, but should be seen as a first step towards standardization of WWW management.
An implementation of this MIB already exists. Due to the use of a commercial development package it cannot be distributed. However, a public domain version is now being developed.

4. HyperText Transfer Protocol MIB structure

The HTTP MIB module contains detailed network management information concerning HTTP. The MIB module is divided into entity- and traffic-related information.

The HTTP MIB consists of three groups:

1. httpSystem,
2. httpStatistics, and
3. httpTimeOuts.

4.1. System group

The System group consists of the httpEntityTable. This table contains not only basic network management information for (potentially) multiple HTTP entities running on a single host machine, but also entity information for virtual domains which give the physical interface of the host multiple addresses. The table is indexed with a unique number specifying the entity in the table which is also used as index of the other tables in this MIB. For each entity the following information is provided and:

- The protocol implemented by the entity; this value should be derived from the assigned number for the service; in the case of HTTP, this would be 80.

- A brief description of the entity and a contact e-mail address for the person responsible.

- The version of the protocol implemented by the entity, the producer of the software, the release version, and (optionally) an object identifier for the producer.

- Configuration information such as the address of the host machine (both in text form and as an IP address), the port in use, the time of last initialization, and whether the entity is a client, server, proxy, or a caching proxy.

4.2. Statistics group

The Statistics group provides network management information for the network traffic received and transmitted by the entity. The group consists of the httpSummaryTable, the httpRequestTable, and the httpResponseTable.

- The SummaryTable contains a set of counters for each HTTP entity which provide a quick summary of the number of requests received, bytes transmitted, and so on. The SummaryTable also holds counters for Requests and Responses which have been discarded or received in error.
  However certain variables are redundant with respect to the Request and Response tables, they are added to reduce network traffic.

- The Request and Response tables provide much more detailed information broken down by entity and the type of Request or Response. For each entity, each type of Request or Response has a separate entry giving a count of the number received, sent, and time stamps for the last interaction.

These tables are not only indexed by an entityIndex pointing to an entry in the httpEntityTable, but also with the Request c.q. Response type. The main reason to define this in a table is that the MIB objects are not dependent on the protocol standard. Only the textual convention should be changed then new types can simply be added by the implementation.

2.3. TimeOuts group

The final group in the HTTP module contains timeout information for each of the HTTP entities. The information is presented in the form of a table which may be resized by the Network Management System, and contains the address of the remote entity and the time at which the timeout occurred.

5. HyperText Transfer Protocol MIB definition

```
HTTP-MIB DEFINITIONS ::= BEGIN

IMPORTS
      MODULE-IDENTITY, OBJECT-TYPE, experimental, Counter32
                  FROM SNMPv2-SMI
      TEXTUAL-CONVENTION, DisplayString, TimeStamp
                  FROM SNMPv2-TC
      MODULE-COMPLIANCE, OBJECT-GROUP
                  FROM SNMPv2-CONF;


httpMIB MODULE-IDENTITY
      LAST-UPDATED"9511080000Z"
      ORGANIZATION"HTTP MIB Interest Group"
      CONTACT-INFO
                  "        Mark Gamble

                  Post:    ESYS Limited
                           Berkeley House
                           London Square
                           Cross Lanes
                           GUILDFORD
                           Surrey
                           UK

                  Tel:     +44 1483 304545
                  Fax:     +44 1483 303878
                  Email:   mgamble@esys.co.uk"
      DESCRIPTION
                "The MIB module for http Servers and Clients. The http
                in the module name is intended to cover a family of
                ‘Networked Information Retrieval' protocols such as
                http, nntp, ftp, gopher and so on.
                  Membership of this family is difficult to define
                exactly, but all members share a similar
                request-response structure used to retrieve information
                (in the form of files, documents, articles) from a
                remote server."
      ::= { enterprises universityOfTwente(785) 2 }

httpMIBObjects OBJECT IDENTIFIER ::= { httpMIB 1 }
httpMIBConformance OBJECT IDENTIFIER ::= { httpMIB 2 }
httpMIBCompliances OBJECT IDENTIFIER ::= { httpMIBConformance 1 }
httpMIBGroups OBJECT IDENTIFIER ::= { httpMIBConformance 2 }
```

```
HttpMethod ::= TEXTUAL-CONVENTION
      STATUS      current
      DESCRIPTION
                  "This data type is used to describe http methods. The
                value of a variable of this type is exactly the same
                method token usedin an http request. The currently
                defined methods for http areGET, HEAD and POST.
                  For ftp, this type would cover the access control,
                   transfer parameter, and service commands."
      SYNTAX      INTEGER {
                        get(1), head(2), put(3), post(4),
                        delete(5), link(6), unlink(7) }

HttpStatusCode ::= TEXTUAL-CONVENTION
      STATUS      current
      DESCRIPTION
                  "The status code of an http response as defined in the
                RFC specification.

                  The StatusCode (or reply code) is structured as a three
                digit code of the attempt to understand and satisfy the
                request.
                  The following description is derived from the HyperText
                  Transfer Protocol RFC:

                  The first digit of the Status-Code defines the class of
                response.
                  The last two digits do not have any categorization role.
                There are 5 values for the first digit:

                  1xx:     Informational - Not used, but reserved for
                        future use.
                  2xx:     Success - The action was successfully received,
                        understood, and accepted.
                  3xx:     Redirection - Further action must be taken in
                        order to complete the request.
                  4xx:     Client Error - The request contains bad syntax
                        or cannot be full filed.
                  5xx:     Server Error - The server failed to fulfill an
                        apparently valid request.

                  Currently defined values for http are:
                  ok(200), created(201), accepted(202), noContent(204),
                  movedPermanently(301), movedTemporarily(302),
                notModified(304), badRequest(400), unauthorized(401),
                  forbidden(403), notFound(404), internalServerError(500),
                  notImplemented(501), badGateway(502),
                serviceUnavailable(503)."

      SYNTAX      INTEGER (100..999)
```

```
--
-- The http System Group
--
-- The http System group contains information about the http
-- protocol entity.
--

httpSystem OBJECT IDENTIFIER ::= { httpMIBObjects 1 }

httpEntityTable OBJECT-TYPE
      SYNTAX          SEQUENCE OF HttpEntityEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The table of http Servers and Clients present on the
                system."
    ::= { httpSystem 1 }

httpEntityEntry OBJECT-TYPE
      SYNTAX          HttpEntityEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "Details of a particular http Server or Client."
      INDEX       { httpEntityIndex }
      ::= { httpEntityTable 1 }

HttpEntityEntry ::= SEQUENCE {
      httpEntityIndex              INTEGER,
      httpEntityProtocol   INTEGER,
      httpEntityDescriptionDisplayString,
      httpEntityContact    DisplayString,
      httpEntityProtocolVersionDisplayString,
      httpEntityVendor     DisplayString,
      httpEntityVersion    DisplayString,
      httpEntityObjectID   OBJECT IDENTIFIER,
      httpEntityAddress    DisplayString,
      httpEntityPort               INTEGER,
      httpEntityIpAddress  IpAddress,
      httpEntityLastInitialisationTimeStamp,
      httpEntityType               INTEGER
       }

httpEntityIndex OBJECT-TYPE
      SYNTAX          INTEGER (1..2147483647)
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "A unique (on this machine) identifier for this entity."
      ::= { httpEntityEntry 1 }
      -- Instrumentation:  Agent internal.
```

```
httpEntityProtocol OBJECT-TYPE
      SYNTAX          INTEGER (1..2147483647)
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "This should be the number from /etc/services (or its
                equivalent) which is associated with the service
                implemented.  For example, the value of this variable
                would be 21 for ftp, 80 for http."
      ::= { httpEntityEntry 2 }
      -- Instrumentation:  System file.

httpEntityDescription OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "Textual description of the http Server or Client."
      ::= { httpEntityEntry 3 }
      -- Instrumentation:  Configuration file.

httpEntityContact OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The textual identification of the contact person
                  for this http Server or Client, together with
                  information on how to contact this person."
      ::= { httpEntityEntry 4 }
      -- Instrumentation:  Configuration file.

httpEntityProtocolVersion OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "Textual description of the version of the protocol
                implemented."
      ::= { httpEntityEntry 5 }
      -- Instrumentation:  Log file

httpEntityVendor OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "Textual description of the organization which
                implemented the protocol."
      ::= { httpEntityEntry 6 }
      -- Instrumentation:  Log file
```

```
httpEntityVersion OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "Textual description of the implementated version."
       ::= { httpEntityEntry 7 }
      -- Instrumentation:  Log file

httpEntityObjectID OBJECT-TYPE
      SYNTAX          OBJECT IDENTIFIER
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The authoritative identification for the private MIB
                for this http Entity, presumably based on the vendor.

                  If no OBJECT IDENTIFIER exists for the private MIB,
                attempts to access this object will return noSuchName
                (SNMPv1) or noSuchInstance (SNMPv2)."
      ::= { httpEntityEntry 8 }
      -- Instrumentation:  Direct access

httpEntityAddress OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The DNS address at which the http Entity listens for
                  Requests or Responses. This is variable is useful
                when the entity listens to a virtual domain (address)."
      ::= { httpEntityEntry 9 }
      -- Instrumentation:  Config file

httpEntityPort OBJECT-TYPE
      SYNTAX          INTEGER (0..4096)
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The TCP port at which the http Entity listens for
                  Requests or Responses."
      ::= { httpEntityEntry 10 }
      -- Instrumentation:  Config file

httpEntityIpAddress OBJECT-TYPE
      SYNTAX          IpAddress
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The IP Address at which the http Entity listens for
                  Requests or Responses."
      ::= { httpEntityEntry 11 }
      -- Instrumentation:  System
```

```
httpEntityLastInitialisation OBJECT-TYPE
      SYNTAX          TimeStamp
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The value of sysUpTime at the time the http Entity
                  was last initialised. If the http Entity was
                  last initialised prior to the last initialisation of the
                  network management subsystem, then this object contains
                  a zero value."
      ::= { httpEntityEntry 12 }
      -- Instrumentation:  Config file

httpEntityType OBJECT-TYPE
      SYNTAX          INTEGER {   server(1), client(2), proxy(3),
                                    cachingProxy(4) }
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "Identification of the role of the http Entity."
      ::= { httpEntityEntry 13 }
      -- Instrumentation:  Direct access

--
-- The http Statistics Group
--
-- The http Statistics group contains information concerning the
-- utilization of the http protocol entity.
--

httpStatistics OBJECT IDENTIFIER ::= { httpMIBObjects 2 }

httpSummaryTable OBJECT-TYPE
      SYNTAX          SEQUENCE OF HttpSummaryEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The table providing overview statistics for the http
                protocol entities on this system."
       ::= { httpStatistics 1 }

httpSummaryEntry OBJECT-TYPE
      SYNTAX          HttpSummaryEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "Overview statistics for an individual http entity."
      INDEX       { httpSummaryEntityIndex }
      ::= { httpSummaryTable 1 }
```

```
HttpSummaryEntry ::= SEQUENCE {
      httpSummaryEntityIndex       INTEGER,
      httpSummaryRequests          Counter32,
      httpSummaryRequestErrorsCounter32,
      httpSummaryRequestDiscardsCounter32,
      httpSummaryResponses         Counter32,
      httpSummaryResponseErrorsCounter32,
      httpSummaryResponseDiscardsCounter32,
      httpSummaryInUnknowns        Counter32,
      httpSummaryInBytes           Counter32,
      httpSummaryOutBytes          Counter32,
      httpSummaryTimeOuts          Counter32
    }

httpSummaryEntityIndex OBJECT-TYPE
      SYNTAX          INTEGER (1..2147483647)
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The unique (on this machine) identifier for this
                 entity. This Index corresponds to httpEntityIndex in
                 the System group."
      ::= { httpSummaryEntry 1 }
      -- Instrumentation:  Agent internal

httpSummaryRequests OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The total number of Requests generated or received by
                 this entity."
      ::= { httpSummaryEntry 2 }
      -- Instrumentation:  Log file

httpSummaryRequestErrors OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The total number of Request errors detected by this
                 entity (server only.)"
      ::= { httpSummaryEntry 3 }
      -- Instrumentation:  Log file

httpSummaryRequestDiscards OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The total number of Requests discarded by this entity
                  (server only)."
      ::= { httpSummaryEntry 4 }
      -- Instrumentation:  Direct Access
```

```
httpSummaryResponses OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The total number of Responses generated or received by
                this entity."
      ::= { httpSummaryEntry 5 }
      -- Instrumentation:  Log file

httpSummaryResponseErrors OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The total number of Response errors detected by this
                entity (client only)."
      ::= { httpSummaryEntry 6 }
      -- Instrumentation:  Direct Access

httpSummaryResponseDiscards OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The total number of Responses discarded by this entity
                  (client only.)"
      ::= { httpSummaryEntry 7 }
      -- Instrumentation:  Direct Access

httpSummaryInUnknowns OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The total number of unknown messages received by this
                entity."
      ::= { httpSummaryEntry 8 }
      -- Instrumentation:  Log file

httpSummaryInBytes OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The total number of bytes received by this entity."
      ::= { httpSummaryEntry 9 }
      -- Instrumentation:  Log file
```

```
httpSummaryOutBytes OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The total number of bytes generated by this entity."
      ::= { httpSummaryEntry 10 }
      -- Instrumentation:  Log file

httpSummaryTimeOuts OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The number of timeouts for this entities."
      ::= { httpSummaryEntry 11 }
      --Instrumentation:  Log file

httpRequestTable OBJECT-TYPE
      SYNTAX          SEQUENCE OF HttpRequestEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The table providing detailed request statistics for
                the http protocol entities on this system."
      ::= { httpStatistics 2 }

httpRequestEntry OBJECT-TYPE
      SYNTAX          HttpRequestEntry
      MAX-ACCESS   not-accessible
      STATUS          current
      DESCRIPTION
                  "Request statistics for an individual http entity."
      INDEX       { httpRequestEntityIndex, httpRequestMethodIndex }
      ::= { httpRequestTable 1 }

HttpRequestEntry ::= SEQUENCE {
      httpRequestEntityIndexINTEGER,
      httpRequestMethodIndexHttpMethod,
      httpRequestInCount   Counter32,
      httpRequestInLastTimeTimeStamp,
      httpRequestOutCount  Counter32,
      httpRequestOutLastTimeTimeStamp
      }
```

```
httpRequestEntityIndex OBJECT-TYPE
      SYNTAX          INTEGER (1..2147483647)
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The unique (on this machine) identifier for this
                entity. This Index corresponds to httpEntityIndex
                in the System group."
      ::= { httpRequestEntry 1 }
      -- Instrumentation:  Agent internal

httpRequestMethodIndex OBJECT-TYPE
      SYNTAX          HttpMethod
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The particular request method the statistics apply to."
      ::= { httpRequestEntry 2 }
      -- Instrumentation:  Log file

httpRequestInCount OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The number of requests of this type received by this
                entity."
      ::= { httpRequestEntry 3 }
      -- Instrumentation:  Log file

httpRequestInLastTime OBJECT-TYPE
      SYNTAX          TimeStamp
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The value of sysUpTime at the time the last request
                was received."
      ::= { httpRequestEntry 4 }
      -- Instrumentation:  Log file

httpRequestOutCount OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The number of requests of this type generated by this
                entity."
      ::= { httpRequestEntry 5 }
      -- Instrumentation:  Log file
```

```
httpRequestOutLastTime OBJECT-TYPE
      SYNTAX          TimeStamp
      MAX-ACCESS      read-only
      STATUS          current
      DESCRIPTION
                  "The value of sysUpTime at the time the last request
                was generated."
      ::= { httpRequestEntry 6 }
      -- Instrumentation:  Log file

httpResponseTable OBJECT-TYPE
      SYNTAX      SEQUENCE OF HttpResponseEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The table providing detailed response statistics for
                the http protocol entities on this system."
      ::= { httpStatistics 3 }

httpResponseEntry OBJECT-TYPE
      SYNTAX               HttpResponseEntry
      MAX-ACCESS       not-accessible
      STATUS           current
      DESCRIPTION
                  "Response statistics for an individual http Server or
                Client."
      INDEX       { httpResponseEntityIndex, httpResponseStatusIndex }
      ::= { httpResponseTable 1 }

HttpResponseEntry ::= SEQUENCE {
      httpResponseEntityIndexINTEGER,
      httpResponseStatusIndexHttpStatusCode,
      httpResponseInCount  Counter32,
      httpResponseInLastTimeTimeStamp,
      httpResponseOutCount Counter32,
      httpResponseOutLastTimeTimeStamp
      }

httpResponseEntityIndex OBJECT-TYPE
      SYNTAX          INTEGER (1..2147483647)
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The unique (on this machine) identifier for this
                entity. This Index corresponds to httpEntityIndex in
                the System group."
      ::= { httpResponseEntry 1 }
      -- Instrumentation:  Agent internal
```

```
httpResponseStatusIndex OBJECT-TYPE
      SYNTAX          HttpStatusCode
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The particular response status the statistics apply
                to."
      ::= { httpResponseEntry 2 }
      -- Instrumentation:  Log file

httpResponseInCount OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
        STATUS          current
      DESCRIPTION
                  "The number of responses of this type received by this
                entity."
      ::= { httpResponseEntry 3 }
      -- Instrumentation:  Log file

httpResponseInLastTime OBJECT-TYPE
      SYNTAX          TimeStamp
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The value of sysUpTime at the time the last response
                was received."
      ::= { httpResponseEntry 4 }
      -- Instrumentation:  Log file

httpResponseOutCount OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The number of responses of this type generated by
                this entity."
      ::= { httpResponseEntry 5 }
      -- Instrumentation:  Log file

httpResponseOutLastTime OBJECT-TYPE
      SYNTAX          TimeStamp
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The value of sysUpTime at the time the last response
                was generated."
      ::= { httpResponseEntry 6 }
      -- Instrumentation:  Log file
```

```
--
-- The Time Out group contains information about the time outs occurred
-- with the protocol entity.
--

httpTimeOuts OBJECT IDENTIFIER ::= { httpMIBObjects 3 }

httpTimeoutTableSize OBJECT-TYPE
      SYNTAX          INTEGER (0..64)
      MAX-ACCESS      read-write
      STATUS          current
      DESCRIPTION
                  "The number of last TimeOuts contained by the
                httpTimeOutTable."
      ::= { httpTimeOuts 1 }
      -- Instrumentation:  Log file

httpTimeoutTable OBJECT-TYPE
      SYNTAX          SEQUENCE OF HttpTimeoutEntry
      MAX-ACCESS      not-accessible
      STATUS          current
      DESCRIPTION
                  "The table providing detailed timeout statistics for
                the http protocol entities on this system."
      ::= { httpTimeOuts 2 }

httpTimeoutEntry OBJECT-TYPE
      SYNTAX          HttpTimeoutEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "Timeout statistics for a particular http entity."
      INDEX       { httpTimeoutEntityIndex, httpTimeoutNumber }
      ::= { httpTimeoutTable 1 }

HttpTimeoutEntry ::= SEQUENCE {
      httpTimeoutEntityIndexINTEGER,
      httpTimeoutNumber    INTEGER,
      httpTimeoutRemoteAddressDisplayString,
      httpTimeoutTime               TimeStamp
      }

httpTimeoutEntityIndex OBJECT-TYPE
      SYNTAX          INTEGER (1..2147483647)
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The unique (on this machine) identifier for this
                entity. This Index corresponds to httpEntityIndex
                in the http System group."
      ::= { httpTimeoutEntry 1 }
      -- Instrumentation:  Agent internal
```

```
httpTimeoutNumber OBJECT-TYPE
      SYNTAX          INTEGER (0..64)
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The unique identifier for the last Timeout."
      ::= { httpTimeoutEntry 2 }
      -- Instrumentation:  Agent internal

httpTimeoutRemoteAddress OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The address of the remote entity."
      ::= { httpTimeoutEntry 3 }
      -- Instrumentation:  Log file

httpTimeoutTime OBJECT-TYPE
      SYNTAX          TimeStamp
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The time when the time out occured with the
                remote entity."
      ::= { httpTimeoutEntry 4 }
      -- Instrumentation:  Log file

--
-- Conformance and compliance definitions.
--

httpMIBCompliance MODULE-COMPLIANCE
      STATUS          current
      DESCRIPTION
                  "The compliance statement for SNMP entities
                  which implement the HTTP MIB."
      MODULE  -- this module
                  MANDATORY-GROUPS { httpMIBGroup }
      ::= { httpMIBCompliances 4 }
```

```
httpMIBGroup OBJECT-GROUP
      OBJECTS {
                httpEntityIndex, httpEntityProtocol,
                httpEntityDescription, httpEntityContact,
                httpEntityProtocolVersion, httpEntityVendor,
                  httpEntityVersion, httpEntityObjectID,
                  httpEntityAddress, httpEntityPort, httpEntityIpAddress,
                  httpEntityLastInitialisation, httpEntityType,
                  httpSummaryEntityIndex,
                  httpSummaryRequests, httpSummaryRequestErrors,
                  httpSummaryRequestDiscards,
                  httpSummaryResponses, httpSummaryResponseErrors,
                  httpSummaryResponseDiscards,
                  httpSummaryInUnknowns,
                  httpSummaryInBytes, httpSummaryOutBytes,
                  httpSummaryTimeOuts,
                  httpRequestEntityIndex,
                  httpRequestMethodIndex, httpRequestInCount,
                  httpRequestInLastTime, httpRequestOutCount,
                  httpResponseEntityIndex,
                  httpResponseStatusIndex, httpResponseInCount,
                  httpResponseInLastTime, httpResponseOutCount,
                  httpResponseOutLastTime,
                  httpTimeoutNumber,
                  httpTimeoutEntityIndex, httpTimeoutRemoteAddress,
                  httpTimeoutTime }
      STATUS      current
      DESCRIPTION
                  "The collection of objects allowing the
                  management of HTTP servers and clients."
      ::= { httpMIBGroups 1 }

END
```

6. Acknowledgments

This document has been produced by the University of Twente (The Netherlands), together with ESYS Limited (The United Kingdom), as part of a ‘proof of concept' study for the ‘Centre of Earth Observation' (CEO) of the ‘Joint Research Centre' (JRC) of the European Community. This document has benefited greatly to the comments of:

Carl W. Kalbfleisch
<cwk@onramp.net>

Mark Gamble
<mgamble@esys1.esys.co.uk>

Rui Meneses
<rui.meneses@jrc.it>

Juergen Schoenwaelder
<schoenw@cs.utwente.nl>

7. References

[1] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1902, January 1996.

[2] McCloghrie, K., and M. Rose, Editors, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", STD 17, RFC 1213, Hughes LAN Systems, Performance Systems International, March 1991.

[3] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Textual Conventions for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1903, January 1996.

[4] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1905, January 1996.

[5] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Conformance Statements for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1904, January 1996.

[6] Case, J., M. Fedor, M. Schoffstall, J. Davin, "Simple Network Management Protocol", RFC 1157, SNMP Research, Performance Systems International, MIT Laboratory for Computer Science, May 1990.

[7] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Introduction to Community-based SNMPv2", RFC 1901, January 1996.

[8] HTTP Working Group, Berners-Lee, T., Fielding R. and Frystyk, H., "Hypertext Transfer Protocol -- HTTP/1.0", Internet draft, October 1995.

8. Security Considerations

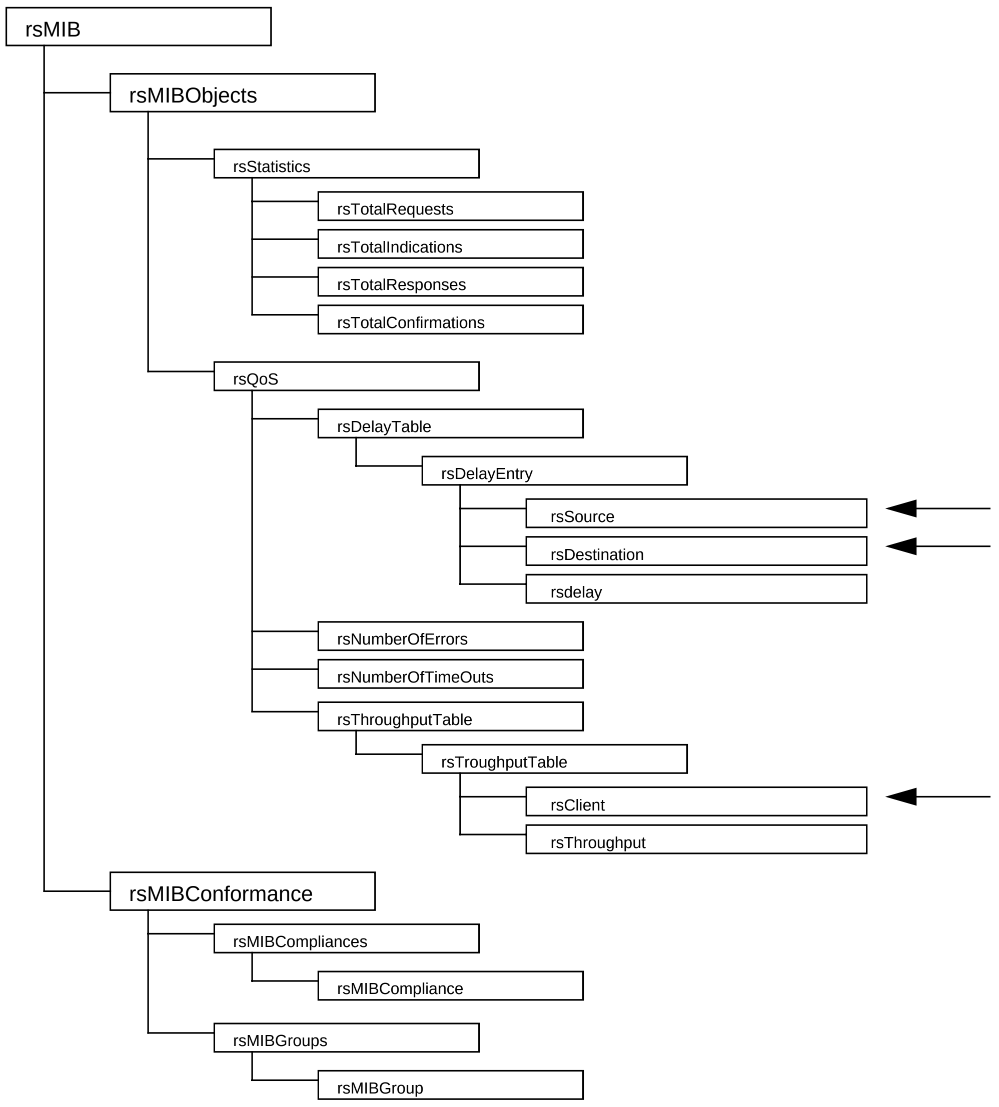
Security issues are not discussed in this memo.

9. Authors' Addresses

Harrie Hazewinkel / Eric van Hengstum / Aiko Pras
University of Twente
Centre for Telematics and Information Technology (CTIT)
POBox 217
7500 AE Enschede, The Netherlands
Phone: +31-53-4893778
Email: hazewink@cs.utwente.nl
       hengstum@cs.utwente.nl
       pras@cs.utwente.nl

Table of Contents

Appendix B: Retrieval Service MIB definition



Definitions of Managed Objects for an Information Retrieval Service

April 22, 1996

<draft-hazewinkel-rsmib-01.txt>

Harrie Hazewinkel
University of Twente
hazewink@cs.utwente.nl

Eric van Hengstum
University of Twente
hengstum@cs.utwente.nl

Aiko Pras
University of Twente
pras@cs.utwente.nl

Status of this Memo

1. Abstract

This memo defines a MIB for use with managing information services. The term "information services" is construed to mean any information providing application, such as World Wide Web (WWW), File Transfer Protocol (FTP), and Gopher. The retrieval service is an abstraction for the information transport protocol. The retrieval service which is a connection-less service can by instantiate by, for instance, the Hyper Text Transfer Protocol, or the File Transfer Protocol.

2. TheSNMPv2 Network Management Framework

The SNMPv2 Network Management Frameworkconsists of four major components. They are:

o   STD 17, RFC 1213 [2] defines MIB-II, the core set of managed objects for the Internet suite of protocols.

o   RFC 1901 Introduction to Community-based SNMPv2

o   RFC 1902 Structureof Management Information for Version 2of the SimpleNetworkManagement Protocol (SNMPv2)

o   RFC 1903 Textual Conventions for Version 2of the Simple Network ManagementProtocol (SNMPv2)

o   RFC 1904 Conformance Statements for Version 2 of the Simple Network ManagementProtocol (SNMPv2)

o   RFC 1907 Management Information Base for Version 2of the Simple Network Management Protocol(SNMPv2)

o   RFC 1908 Coexistence between Version 1 andVersion2 of the Internet-standard Network Management Framework

The Framework permits new objects to bedefinedfor thepurposeof experimentationand evaluation.

2.1. Object Definitions

Managedobjects   are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the subset ofAbstract SyntaxNotation One (ASN.1)defined   in the SMI[1]. In particular, each object type is named by an OBJECT IDENTIFIER, an administratively assigned name. The object typetogether with an objectinstance servesto uniquely identify a specific instantiation of the object. For human convenience, we often usea textual string, termed the object descriptor, to refer to the object type.

3. Introduction.

The Retrieval Service is a connection-less service providing transport of information between a client and a server. The service is an abstraction of the information transport protocol used.
The Retrieval Service MIB module contains network management information about a provided service. Because the Retrieval Service is an abstraction of the information transport service used in, for instance, the World Wide Web, details of the transport protocol providing this service are of no importance to the users of the Retrieval Service.

The work performed for this MIB is a result of a project executed for the Centre of Earth Observations. It is not seen as complete, but it should be a first step in an effort to manage the WWW application. An implementation of this MIB already exists. Due to the use of a commercial development package it cannot be distributed. However, a public domain version is now developed.

4. Retrieval Service MIB structure

The Retrieval Service module contains the following groups:
1. service primitives/ statistics,
2. quality of service.

4.1. Statistics group

Communication via the Retrieval Service takes place by means of service primitives used by the connection-less service. The statistics group contains statistical information concerning the service primitives passed over the Service Access Point (SAP). The variables of this group count the number of service primitives executed on the retrieval service.

Two possible solutions were seen to define these counters:
1. The total numbers of service primitives are counted by type. For each service primitive type used by the retrieval service a managed object has to be defined.
2. The total number of service primitives related to a remote host and service primitive-type are counted. This results in a conceptual table; for each service primitive type used by the retrieval service a table has to be defined, indexed with the IpAddress of the remote host. The remote host is a parameter of the service primitive.

It was decided that the simplicity of the first solution outweighed the increased information present in the second.

4.2. Quality of Service group

The Quality of Service QoS group contains network management information about the quality of the retrieval service. The information in this group provides the network manager with visibility of the performance of the underlying network.

The recognized QoS parameters are:

- The transport delay is defined as a table. The table is indexed with the source and destination addresses of the delay.

  The total round trip delay can be computed by the Network Management System user.

- The number of errors variable provides information on errors which have occurred in the retrieval service.
- The number of timeouts variable provides information on timeouts which have occurred in the retrieval service.
- The throughput table provides information on the speed of the network with a given client. The address of the client is used to index the table.

5.   Retrieval Service MIB definition

```
RS-MIB DEFINITIONS ::=
BEGIN

IMPORTS
      enterprises, MODULE-IDENTITY, OBJECT-TYPE, Counter32, TimeTicks,
      IpAddress
                  FROM SNMPv2-SMI
      MODULE-COMPLIANCE, OBJECT-GROUP
                  FROM SNMPv2-CONF;

rsMIB MODULE-IDENTITY
      LAST-UPDATED"9601251800Z"
      ORGANIZATION"University Of Twente"
      CONTACT-INFO
                  "        Harrie Hazewinkel
                  Postal:  Centre of Telematics and
                                      Information Technology
                           University Of Twente
                           POBox 217
                           7500 AE Enschede
                           The Netherlands
                  phone :  +31 53 8943746
                  E-mail:  H.Hazewinkel@cs.utwente.nl

                  This document benefited greatly from the comments of
                  all participants in the CEO project for management of
                  World Wide Web Servers."
      DESCRIPTION
                  "A MIB module for the retrieval service. The retrieval
                  service is a service providing the capability of
                  information transport via a network.

                  It provides management information about an
                  information transport, which is exactly describing
                  what the transport provider is doing for the user,
                  without showing detailed information from inside
                  the transport provider, HTTP."
      ::= { enterprises universityOfTwente(785) 3 }

rsMIBObjects OBJECT IDENTIFIER ::= { rsMIB 1 }
rsMIBConformance OBJECT IDENTIFIER ::= { rsMIB 2 }
rsMIBCompliances OBJECT IDENTIFIER ::= { rsMIBConformance 1 }
rsMIBGroups OBJECT IDENTIFIER ::= { rsMIBConformance 2 }
```

```
--
-- The service statistics provides management
-- information about the service primitives that are
-- executed on the HTTP Service Provider.
--

rsStatistics OBJECT IDENTIFIER ::= { rsMIBObjects 1 }

rsTotalRequests OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The total number of requests that have been executed
                  on the HTTP service provider.

                  This field is only interesting when management
                  functions are implemented at the client side."
      ::= { rsStatistics 1 }
      -- Instrumentation: client

rsTotalIndications OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The total number of indications that have been executed
                  on the HTTP service provider."
      ::= { rsStatistics 2 }
      -- Instrumentation: logfile

rsTotalResponses OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The total number of responses that have been executed
                  on the HTTP service provider."
      ::= { rsStatistics 3 }
      -- Instrumentation: logfile

rsTotalConfirmations OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The total number of confirmations that have been
                  executed on the HTTP service provider.

                  This field is only interesting when management
                  functions are implemented at the client side."
      ::= { rsStatistics 4 }
      -- Instrumentation: client
      --
      -- The provided QoS of the service provider.
      --

rsQoS OBJECT IDENTIFIER ::= { rsMIBObjects 2 }

rsDelayTable OBJECT-TYPE
      SYNTAX          SEQUENCE OF RsDelayEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The table of the delay between source and destination."
      ::= { rsQoS 1 }

rsDelayEntry OBJECT-TYPE
      SYNTAX          RsDelayEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "Details of a particular http Server or Client."
      INDEX       { rsSource, rsDestination }
      ::= { rsDelayTable 1 }

RsDelayEntry ::=
        SEQUENCE {
      rsSource        DisplayString,
      rsDestination   DisplayString,
      rsDelay         TimeInterval
      }

rsSource OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The DNS name of the source."
      ::= { rsDelayEntry 1 }
      -- Instrumentation: logfile / own system

rsDestination OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The DNSname of the destination."
      ::= { rsDelayEntry 2 }
      -- Instrumentation: logfile / own system
```

```
rsDelay OBJECT-TYPE
      SYNTAX          TimeInterval
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The delay which occured during transport of information
                  from source to destination."
      ::= { rsDelayEntry 3 }
      -- Instrumentation: client / server

rsNumberOfErrors OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The number of errors that have been occured."
      ::= { rsQoS 2 }
      -- Instrumentation: client

rsNumberOfTimeouts OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The number of time-outs that have been occured."
      ::= { rsQoS 3 }
      -- Instrumentation: logfile

rsTroughputTable OBJECT-TYPE
      SYNTAX          SEQUENCE OF RsTroughputEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The table of the throughput with a certain client."
      ::= { rsQoS 4 }

rsTroughputEntry OBJECT-TYPE
      SYNTAX          RsTroughputEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "Details of a particular http Server or Client."
      INDEX       { rsClient }
      ::= { rsTroughputTable 1 }

RsTroughputEntry ::=
        SEQUENCE {
      rsClient             DisplayString,
      rsThroughput         INTEGER
      }
```

```
rsClient OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The DNS name of the client."
      ::= { rsTroughputEntry 1 }
      -- Instrumentation: logfile

rsThroughput OBJECT-TYPE
      SYNTAX          INTEGER (1..2147483647)
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The throughput of data with the client."
      ::= { rsTroughputEntry 2 }
      -- Instrumentation: direct access

--
-- Conformance and compliance definitions.
--

rsMIBCompliance MODULE-COMPLIANCE
      STATUS          current
      DESCRIPTION
                  "The compliance statements for http service
                  which implement the HTTP MIB"
      MODULE
                  MANDATORY-GROUPS { rsGroup }
      ::= { rsMIBCompliances 1 }

rsMIBGroup OBJECT-GROUP
      OBJECTS {   rsTotalRequests, rsTotalIndications,
                  rsTotalResponses, rsTotalConfirmations,
                  rsSource, rsDestination, rsDelay,
                  rsNumberOfErrors,
                  rsNumberOfTimeouts,
                  rsClient,rsThroughput }
      STATUS      current
      DESCRIPTION
                  "The rsGroup defines the objects
                  of the retrieval service."
      ::= { rsMIBGroups 1 }

END
```

6. Acknowledgments

This document has been produced by the University of Twente (The Netherlands), together with ESYS Limited (The United Kingdom), as part of a ‘proof of concept' study for the ‘Centre of Earth Observation' (CEO) of the ‘Joint Research Centre' (JRC) of the European Community. This document has benefited greatly to the comments of:

Mark Gamble
<mgamble@esys1.esys.co.uk>

Rui Meneses
<rui.meneses@jrc.it>

Juergen Schoenwaelder
<schoenw@cs.utwente.nl>

7. References

[1] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1902, January 1996.

[2] McCloghrie, K., and M. Rose, Editors, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", STD 17, RFC 1213, Hughes LAN Systems, Performance Systems International, March 1991.

[3] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Textual Conventions for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1903, January 1996.

[4] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1905, January 1996.

[5] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Conformance Statements for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1904, January 1996.

[6] Case, J., M. Fedor, M. Schoffstall, J. Davin, "Simple Network Management Protocol", RFC 1157, SNMP Research, Performance Systems International, MIT Laboratory for Computer Science, May 1990.

[7] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Introduction to Community-based SNMPv2", RFC 1901, January 1996.

8. Security Considerations

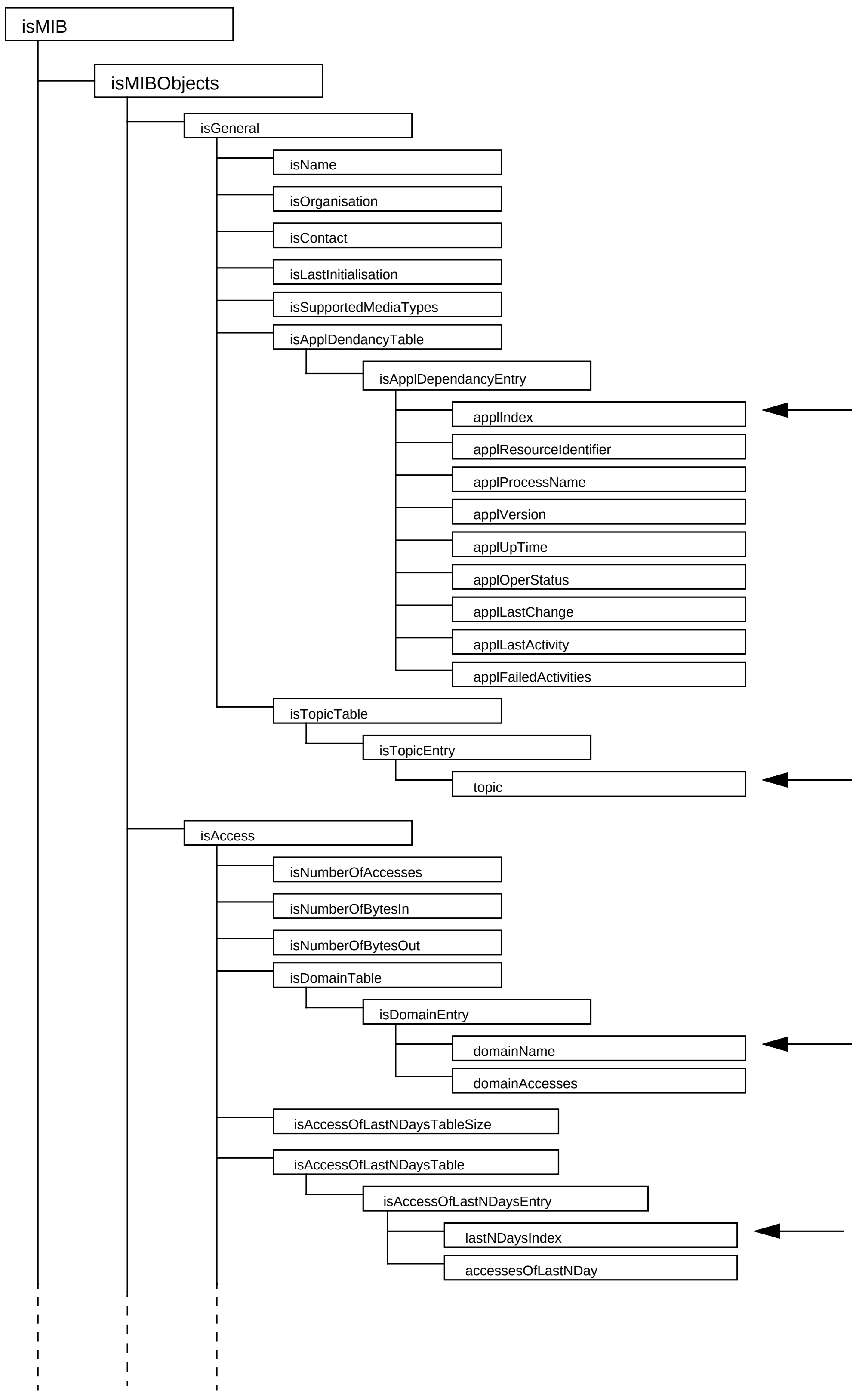
Security issues are not discussed in this memo.

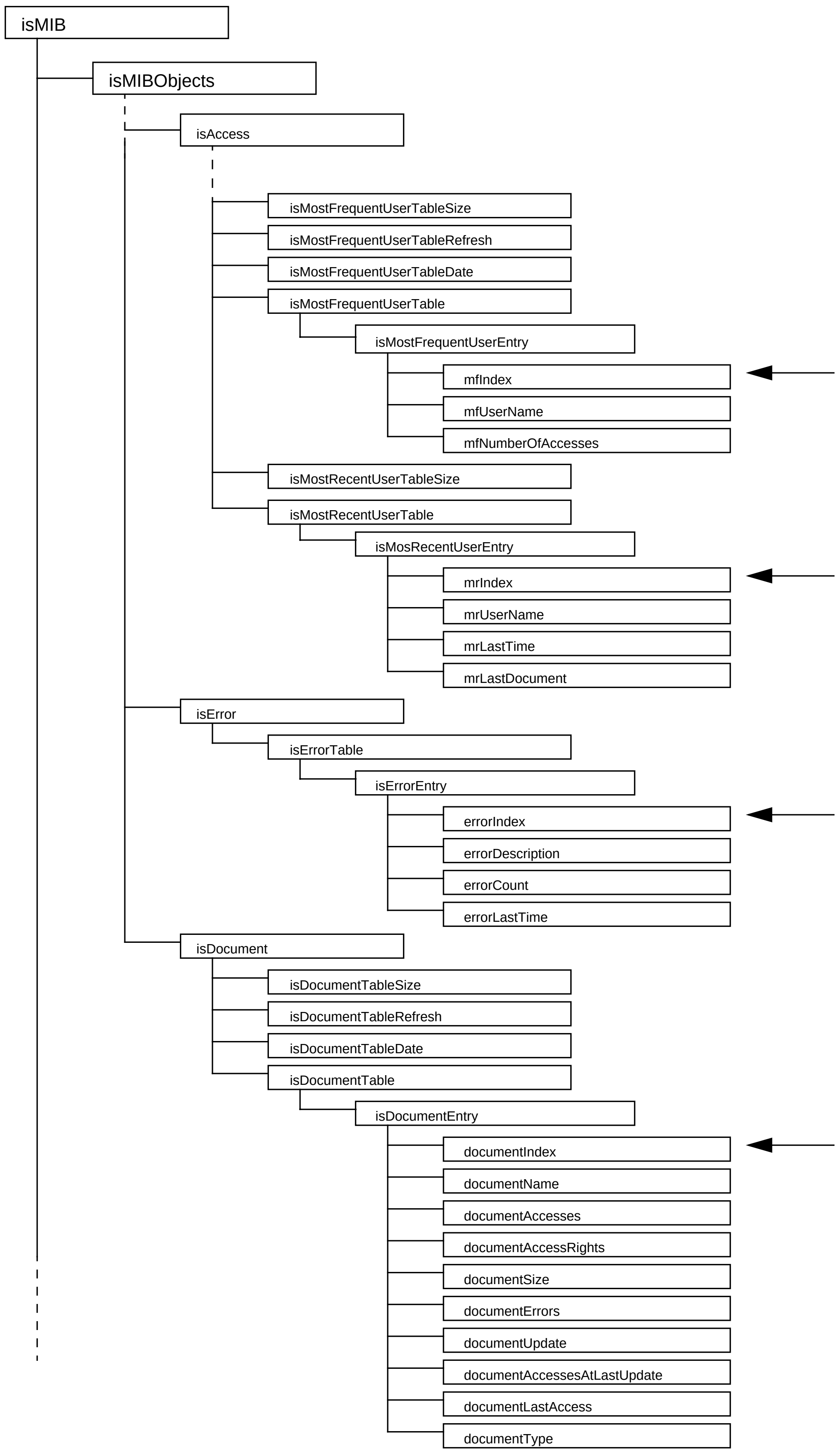
9. Authors' Addresses

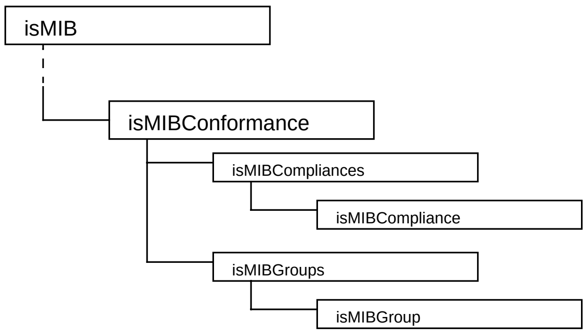
Harrie Hazewinkel / Eric van Hengstum / Aiko Pras
University of Twente
Centre for Telematics and Information Technology (CTIT)
POBox 217
7500 AE Enschede, The Netherlands
Phone: +31-53-4893778
Email: hazewink@cs.utwente.nl
       hengstum@cs.utwente.nl
       pras@cs.utwente.nl

Table of Contents

Appendix C: Information Store MIB definition




isMIB
isMIBObjects
isAccess
isMostFrequentUserTableSize
isMostFrequentUserTableRefresh
isMostFrequentUserTableDate
isMostFrequentUserTable
isMostFrequentUserEntry
mfIndex
mfUserName
mfNumberOfAccesses
isMostRecentUserTableSize
isMostRecentUserTable
isMosRecentUserEntry
mrIndex
mrUserName
mrLastTime
mrLastDocument
isError
isErrorTable
isErrorEntry
errorIndex
errorDescription
errorCount
errorLastTime
isDocument
isDocumentTableSize
isDocumentTableRefresh
isDocumentTableDate
isDocumentTable
isDocumentEntry
documentIndex
documentName
documentAccesses
documentAccessRights
documentSize
documentErrors
documentUpdate
documentAccessesAtLastUpdate
documentLastAccess
documentType


isMIB
isMIBConformance
isMIBCompliances
isMIBCompliance
isMIBGroups
isMIBGroup

Definitions of Managed Objects for an Information Store

April 22, 1996

<draft-hazewinkel-ismib-00.txt>

Harrie Hazewinkel
University of Twente
hazewink@cs.utwente.nl

Eric van Hengstum
University of Twente
hengstum@cs.utwente.nl

Aiko Pras
University of Twente
pras@cs.utwente.nl

Status of this Memo

1. Abstract

This draft defines a MIB for use with managing information stores. The term "information store" is construed to mean any information providing application, such as World Wide Web (WWW), File Transfer Protocol (FTP), and Gopher. The information store is seen as the application responsible for the information providing capabilities.

2. TheSNMPv2 Network Management Framework

The SNMPv2 Network Management Frameworkconsists of four major components. They are:

- STD 17, RFC 1213 [2] defines MIB-II, the core set of managed objects for the Internet suite of protocols.

- RFC 1901 Introduction to Community-based SNMPv2

- RFC 1902 Structureof Management Information for Version 2of the SimpleNetworkManagement Protocol (SNMPv2)

- RFC 1903 Textual Conventions for Version 2of the Simple Network ManagementProtocol (SNMPv2)

- RFC 1904 Conformance Statements for Version 2 of the Simple Network ManagementProtocol (SNMPv2)

- RFC 1907 Management Information Base for Version 2of the Simple Network Management Protocol(SNMPv2)

- RFC 1908 Coexistence between Version 1 andVersion2 of the Internet-standard Network Management Framework

The Framework permits new objects to bedefinedfor thepurposeof experimentationand evaluation.

2.1. Object Definitions

Managedobjects   are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the subset ofAbstract SyntaxNotation One (ASN.1)defined   in the SMI[1]. In particular, each object type is named by an OBJECT IDENTIFIER, an administratively assigned name. The object typetogether with an objectinstance servesto uniquely identify a specific instantiation of the object. For human convenience, we often usea textual string, termed the object descriptor, to refer to the object type.

3. Introduction.

The Information Store represents the information part of a World Wide Web application. The name Information Store (IS) is strongly related to the use of this part of the WWW application. Most WWW servers provide information to users / customers (clients) of the WWW application. In order to maintain and manage the provided information a specific MIB module is defined. The Information Store MIB module specifies the network management information of the WWW server, providing the aforementioned information.

The work performed for this MIB is a result of a project executed for the Centre of Earth Observations. It is not seen as complete, but it should be a first step in an effort to manage the WWW. An implementation of this MIB already exists. Due to the use of a commercial development package it cannot be distributed. However, a public domain version is now being developed.

4. Information Store MIB structure.

The Information Store MIB module structure contains the following groups:

1. isGeneral,
2. isAccess,
3. isErrors, and
4. isDocuments.

4.1 General Group

The isGeneral group contains overall administrative data which uniquely identifies the IS, and should be static for the Information Store (IS). The network management information consists of the administrative data uniquely identifying the IS. In addition to the static data, such as the isContact object (a contact address for the person responsible for the information provided by the IS), the isGeneral group also contains the isTopicTable. The NMS user can use this table to discover what kinds of information are provided by the IS. It was originally intended to include an isLocation variable in this group. This variable would have provided information on the location of the IS, however, as this information is also contained in the sysLocation of the system group it was omitted.

The general group contains also information of applications on which the IS depends (such as database applications for search facilities). The application dependency table provides that information. The status of these applications is checked dynamically and the values are related to the ‘process status' command of the UNIX environment.

4.2 Access group

The access group contains network management information about the network activity of the Information Store. The network activity is described in an abstract manner in regard to the service which transports the information.

This group provides network management information with the following points of view:
1. general data containing totals for the information store,
2. domain data containing totals ordered by domains,
3. daily access data which contains totals for the last N days,
4. user data which contains information concerning the most frequent and most recent users.

The general access data provides statistics relating to usage of the IS. For example, the variables number of accesses (isNumberOfAccesses) and the number of bytes in/ out (isNumberOfBytesOut, isNumberOfBytesIn) are defined.

The isDomainTable breaks down accesses to the IS by Internet domain. The NMS operator can thus discover the countries or groups regular users of the IS belong. This on may be valuable, as an IS administrator can decide whether mirror servers in certain domains may be useful. Then long distance network traffic as well as the server load could be reduced.

To see the number of hits for each day an isAccessOfLastNDaysTable is defined. Together with a variable setting the total number of days which can be traced backwards, this table provides information on the number of accesses executed on the IS on a day to day basis.

The isDomainTable and isAccessOfLastNDaysTable provide useful information on the number and sources of accesses to the IS, however, it is equally important to know what information is attracting Users to the IS. In support of this function, two tables are defined. These tables are a most recent users table and a most frequent users table. The combination of these tables should give meaningful information of users and which information they access.

The isMostFrequentUser table is defined in a slightly different fashion from the isMostRecentUser table. This is mainly because of implementation aspects such as the amount of memory used to store information relating to the IS users and the processing time to generate the information. Details on every User who has accessed the IS must be stored, because new accesses may give that User a new place in the table. Therefore, the following solution was adopted: the table is updated on initiative of the NMS, thus making the NMS responsible for limiting the number of updates requested.

The NMS requests an update by setting the refresh variable to the value ‘false'. An Agent subprocess then processes the new information and, on exiting, sets the refresh variable to the value ‘true'. The subprocess also updates the time stamp variable indicating the date and time when the table was last refreshed.

Besides these variables, the size of the table can also be set with a size variable which has a value in the range of 0 to 128.
The isMostRecentUsers table provides information on which users are the last ones who accessed the IS. The size of the table is limited to at most 128 and can be set in a range of 0 to 128. The table is indexed with the position of the User in the table, and contains information giving the Users name, the time of last access, and document that was accessed.

4.3 Error Group

The error group contains network management information about errors which occurred during accesses of the IS. This group abstracts from the network errors that occur, it only shows errors resulting from information access. This group is defined as a table with an error description, the number of occurrences of that error and the time at which the error last occurred. The reason for defining this information in a table is that an implementor of the MIB module can simply add new errors to the table without changing the MIB module definition.

4.4 Document Group

The document group contains the network management information about the data which is provided by the Information Store. The information is kept in a table containing the documents name, access rights, size, type, and other features such as the number of accesses, time of last update, and any associated errors. The isDocumentTable has the same implementation constraints as the isMostFrequentNUserTable, and is handled in a similar fashion, with TableSize, TableRefresh, and TableData variables. It should be noted that (in the current implementation) e isDocumentTable is populated from User accesses, and thus documents which have not been accessed will not appear in the table.

5.   Information Store MIB definitions

```
IS-MIB DEFINITIONS ::=
BEGIN

IMPORTS
      MODULE-IDENTITY, OBJECT-TYPE, Counter32, TimeTicks, Integer32
                  FROM SNMPv2-SMI
      TEXTUAL-CONVENTION, DisplayString, DateAndTime, TruthValue, TimeStamp
                  FROM SNMPv2-TC
      MODULE-COMPLIANCE, OBJECT-GROUP
                  FROM SNMPv2-CONF;

isMIB MODULE-IDENTITY
      LAST-UPDATED"9601251800Z"
      ORGANIZATION"University Of Twente"
      CONTACT-INFO
                  "        Harrie Hazewinkel

                  Postal:  Centre of Telematics and
                                    Information Technology
                           University Of Twente
                           POBox 217
                           7500 AE Enschede
                           The Netherlands
                  phone :  +31 53 4893746
                  E-mail:  H.Hazewinkel@cs.utwente.nl

                  This document benefited greatly from the comments of
                  all participants in the CEO project for management of
                  World Wide Web Servers."
      DESCRIPTION
                  "A MIB module for an Information Store.
                  An Information Store is the application from which
                  the information is retrieved on the World Wide Web."
      ::= { enterprises universityOfTwente(785) 4 }

isObjects OBJECT IDENTIFIER ::= { isMIB 1 }
isConformance OBJECT IDENTIFIER ::= { isMIB 2 }
isCompliances OBJECT IDENTIFIER ::= { isConformance 1 }
isGroups OBJECT IDENTIFIER ::= { isConformance 2 }
```

```
MediaType ::= TEXTUAL-CONVENTION
      STATUS      current
      DESCRIPTION
                  "This type should be representing the media types
                  which the information store makes available. The
                  media types are those recognized by the RFC 1700.

                  The value is a sum and is initially the value zero.
                  Then, for each media type, L, in the range 1 through
                  7, 2 raised to (L - 1) is added to the sum.
                  For example, a server which only supports video
                  would have a value of 64 (2^(7-1)).  In
                  contrast, a server which is offering text and images
                  would have a value of 17 (2^(1-1) + 2^(5-1)).

                  Note that in the context of this media type,
                  values should be calculated accordingly:
                           1  text
                           2  multipart
                           3  message
                           4  application
                           5  image
                           6  audio
                           7  video"
      SYNTAX INTEGER (0..127)

ApplResourceIdentifier ::= TEXTUAL-CONVENTION
      STATUS      current
      DESCRIPTION
                  "A distinguished name which contains a host on which
                  the application is running and the application's name."
      SYNTAX DisplayString

AccessRights ::= TEXTUAL-CONVENTION
      STATUS      current
      DESCRIPTION
                  "A type with which access rights of a document is
                described.

                  The rights are:
                           read-only permission
                           read-write permission
                           execute permission"
      SYNTAX      INTEGER {
                        read-only(1), read-write(2), execute-able(3)
                        }
```

```
isGeneral OBJECT IDENTIFIER ::= { isObjects 1 }

isName OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS      current
      DESCRIPTION
                  "An administratively-assigned name for this
                  Information Store."
      ::= { isGeneral 1 }
      -- Instrumentation : Config file

isOrganisation OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS      current
      DESCRIPTION
                  "The textual identification of the organisation
                  responsible for the Information Store server.

                  For example, if the organisation which is offering
                  information of the Information Store is named
                  ‘Flintstones, Inc'. This value should have the
                  value of ‘Flintstones, Inc'."
      ::= { isGeneral 2 }
      -- Instrumentation : Config file

isContact OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS      current
      DESCRIPTION
                  "The textual identification of the contact person
                  for this Information Store server, together with
                  information on how to contact this person.

                  For example, when ‘Fred' of the ‘Flintstones, Inc'
                  is maintaining this Information Store, the value
                  should be ‘Fred@Flintstones'."
      ::= { isGeneral 3 }
      -- Instrumentation : Config file

isLastInitialisation OBJECT-TYPE
      SYNTAX          TimeStamp
      MAX-ACCESS  read-only
      STATUS      current
      DESCRIPTION
                  "The value of sysUpTime at the time the Information
                  Store was last initialized. If the Information Store
                  was last initialized prior to the initialization of
                  the network management subsystem, then this object
                  contains a zero value."
      ::= { isGeneral 4 }
      -- Instrumentation : Config file
```

```
isSupportedMediaTypes OBJECT-TYPE
      SYNTAX          MediaType
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "A value which indicates the set of media types that
                  this Information Store primarily offers."
      ::= { isGeneral 5 }
      -- Instrumentation : Config file

isApplDependancyTable OBJECT-TYPE
      SYNTAX          SEQUENCE OF IsApplDependancyEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The table holding objects which apply to all different
                  kinds of applications provided by the Information
                  Store."
      ::= { isGeneral 6 }
      -- Instrumentation : Config file

isApplDependancyEntry OBJECT-TYPE
      SYNTAX          IsApplDependancyEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "An entry associated with an application."
      INDEX {applIndex}
      ::= {isApplDependancyTable 1}

IsApplDependancyEntry ::= SEQUENCE {
      isApplIndex                  INTEGER,
      isApplResourceIdentifierApplResourceIdentifier,
      isApplProcessName    DisplayString,
      isApplVersion                DisplayString,
      isApplUptime                 TimeStamp,
      isApplOperStatus     INTEGER,
      isApplLastChange     TimeStamp,
      isApplLastActivity   TimeStamp,
      isApplFailedActivitiesCounter32
      }

isApplIndex OBJECT-TYPE
      SYNTAX          INTEGER (1..2147483647)
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "An index to uniquely identify the application
                  application."
      ::= {isApplDependancyEntry 1}
      -- Instrumentation : Config file
```

```
isApplResourceIdentifier OBJECT-TYPE
      SYNTAX          ApplResourceIdentifier
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The distinguished name of the host on which the
                application is running and the application name."
      ::= {isApplDependancyEntry 2}
      -- Instrumentation : Config file

isApplProcessName OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The process name of the application."
      ::= {isApplDependancyEntry 3}
      -- Instrumentation : Config file

isApplVersion OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The version of the application software."
      ::= {isApplDependancyEntry 4}
      -- Instrumentation : Config file

isApplUptime OBJECT-TYPE
      SYNTAX          TimeTicks
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The value of sysUpTime at the time the 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."
      ::= {isApplDependancyEntry 5}
      -- Instrumentation : Config file
```

```
isApplOperStatus OBJECT-TYPE
      SYNTAX          INTEGER {running(1), sleeping(2),
                                        runnable(3), idle(4),
                                        zombie(5), traced(6),
                                          sxbrkState(7), notFound(8) }
      MAX-ACCESS      read-only
      STATUS          current
      DESCRIPTION
                 "Indicates the operational status of the network service
                 application. The values are related to the output
                 of the ‘ps'-command and the values are:

                         running: running on a processor.
                         sleeping:waiting for an event to complete.
                         runnable:on run queue.
                         idle:    being created.
                         zombie:  terminated and parent not
                                      waiting.
                         traced:  stopped by a signal because
                                      parent is tracing it.
                         sxbrkState:waiting for more primary memory.
                         notFound:not found in process list."
      ::= {isApplDependancyEntry 6}
      -- Instrumentation : Config file

isApplLastChange OBJECT-TYPE
      SYNTAX          TimeStamp
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                 "The value of sysUpTime at the time the application
                 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."
      ::= {isApplDependancyEntry 7 }
      -- Instrumentation : Config file

isApplLastActivity OBJECT-TYPE
      SYNTAX          TimeStamp
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                 "The value of sysUpTime at the time this application
                 last performed some activity. If the last activity
                 occurred prior to the last initialization of the
                 application, then this object contains a zero value."
      ::= {isApplDependancyEntry 8 }
      -- Instrumentation : Config file
```

```
isApplFailedActivities OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The total number activities where the application
                  has failed, since application initialization."
      ::= {isApplDependancyEntry 9}
      -- Instrumentation : Config file

isTopicTable OBJECT-TYPE
      SYNTAX          SEQUENCE OF IsTopicEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The table holding topics provided by this
                  Information Store."
      ::= { isGeneral 7 }
      -- Instrumentation : Config file

isTopicEntry OBJECT-TYPE
      SYNTAX          IsTopicEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "An entry associated with the topics."
      INDEX       { IMPLIED topic }
      ::= {isTopicTable 1}
      -- Instrumentation : Config file

IsTopicEntry ::= SEQUENCE {
      isTopic             DisplayString
      }

isTopic OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "A textual description of a topic."
      ::= {isTopicEntry 1}

--
-- This group provides general access information and access information
-- organized by domain, day, most recent users, most frequent users.
--

isAccess OBJECT IDENTIFIER ::= { isObjects 2 }
```

```
isNumberOfAccesses OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The total number accesses done at the Information
                  Store."
      ::= { isAccess 1 }
      -- Instrumentation : Log file

isNumberOfBytesOut OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The total number bytes transfered by the Information
                  Store."
      ::= { isAccess 2 }
      -- Instrumentation : Log file

isNumberOfBytesIn OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The total number bytes received by the Information
                  Store."
      ::= { isAccess 3 }
      -- Instrumentation : Direct access

isDomainTable OBJECT-TYPE
      SYNTAX          SEQUENCE OF IsDomainEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The domain table keeps track of the number of
                  accesses done from a specific domain."
      ::= { isAccess 4 }
      -- Instrumentation : Log file

isDomainEntry OBJECT-TYPE
      SYNTAX          IsDomainEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "Entry (conceptual row) of the domain table."
      INDEX       { IMPLIED domainName }
      ::= { isDomainTable 1 }

IsDomainEntry ::= SEQUENCE {
      isDomainName         DisplayString,
      isDomainAccessesCounter32
      }
```

```
isDomainName OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The domain of which the number of accesses
                  specific has to be monitored."
      ::= { isDomainEntry 1 }
      -- Instrumentation : Log file

isDomainAccesses OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The number of accesses done from a certain domain."
      ::= { isDomainEntry 2 }
      -- Instrumentation : Log file

isAccessOfLastNDaysTableSize OBJECT-TYPE
      SYNTAX          INTEGER (0..32)
      MAX-ACCESS  read-write
      STATUS          current
      DESCRIPTION
                  "The number of entries which is contained
                  in the accessOfLastNDaysTable."
      ::= { isAccess 5 }
      -- Instrumentation : Agent internal

isAccessOfLastNDaysTable OBJECT-TYPE
      SYNTAX          SEQUENCE OF IsAccessOfLastNDaysEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The access of last days table keeps track of the
                  number of accesses that have been done in the past.
                  Each entry is a specific day in the past. The day
                  which is specified is related to the current day.

                  This table should be a Top N table and should be
                  configured by the NMS."
      ::= { isAccess 6 }

isAccessOfLastNDaysEntry OBJECT-TYPE
      SYNTAX          IsAccessOfLastNDaysEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "Entry (conceptual row) of the accessOfLastNDaysTable."
      INDEX { lastNDayIndex }
      ::= { accessOfLastNDaysTable 1 }
```

```
IsAccessOfLastNDaysEntry ::= SEQUENCE {
      isLastNDayIndex      INTEGER,
      isAccessOflastNDayCounter32
      }

isLastNDayIndex OBJECT-TYPE
      SYNTAX          INTEGER (0..32)
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The index of the Last N Days Table"
      ::= { accessOfLastNDaysEntry 1 }
      -- Instrumentation : Agent internal

isAccessOflastNDay OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The number of accesses done on the specified day
                  in the past."
      ::= { accessOfLastNDaysEntry 2 }
      -- Instrumentation : logfile

isMostFrequentNUserTableSize OBJECT-TYPE
      SYNTAX          INTEGER (0..128)
      MAX-ACCESS  read-write
      STATUS          current
      DESCRIPTION
                  "The number of entries which is contained
                  in the MostFrequentNUserTable."
      ::= { isAccess 7 }
      -- Instrumentation : Agent internal

isMostFrequentNUserTableRefresh OBJECT-TYPE
      SYNTAX          TruthValue
      MAX-ACCESS  read-write
      STATUS          current
      DESCRIPTION
                  "Field with which a new ‘most frequent N user table'
                  will be made."
      ::= { isAccess 8 }
      -- Instrumentation : Agent internal

isMostFrequentNUserTableDate OBJECT-TYPE
      SYNTAX          DateAndTime
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The universal date and time for the last time the
                  isMostFrequentNUserTable was refreshed.."
      ::= { isAccess 9 }
      -- Instrumentation : Agent internal
```

```
isMostFrequentNUserTable OBJECT-TYPE
      SYNTAX          SEQUENCE OF IsMostFrequentNUserEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The access of document table keeps track of the
                  number of accesses that have been done in the past.
                  Each entry is a separate user.

                  This is a Top N table with the N most recently accessed
                  documents."
      ::= { isAccess 10 }

isMostFrequentNUserEntry OBJECT-TYPE
      SYNTAX          IsMostFrequentNUserEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "Entry (conceptual row) of the document table."
      INDEX       { mfIndex }
      ::= { isMostFrequentNUserTable 1 }

IsMostFrequentNUserEntry ::= SEQUENCE {
      isMfIndex                    INTEGER,
      isMfUserName         DisplayString,
      isMfNumberOfAccessesCounter32
      }

isMfIndex OBJECT-TYPE
      SYNTAX          INTEGER (0..128)
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The index indicates the position in the Top N table."
      ::= { isMostFrequentNUserEntry 1 }
      -- Instrumentation : Agent internal

isMfUserName OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The name of the user."
      ::= { isMostFrequentNUserEntry 2 }
      -- Instrumentation : Log file
```

```
isMfNumberOfAccesses OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The number of accesses made by the user on
                   this server."
      ::= { isMostFrequentNUserEntry 3 }
      -- Instrumentation : Log file

isMostRecentNUserTableSize OBJECT-TYPE
      SYNTAX          INTEGER (0..128)
      MAX-ACCESS  read-write
      STATUS          current
      DESCRIPTION
                  "The number of entries which is contained
                  in the isMostRecentNUserTable."
      ::= { isAccess 11 }
      -- Instrumentation : Agent internal

isMostRecentNUserTable OBJECT-TYPE
      SYNTAX          SEQUENCE OF IsMostRecentNUserEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The access of document table keeps track of the
                  number of accesses that have been done in the past.
                  Each entry is a document.

                  This is a Top N table with the N most recently accessed
                  documents."
      ::= { isAccess 12 }

isMostRecentNUserEntry OBJECT-TYPE
      SYNTAX          IsMostRecentNUserEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "Entry (conceptual row) of the most recent users table."
      INDEX { mrIndex }
      ::= { isMostRecentNUserTable 1 }

IsMostRecentNUserEntry ::= SEQUENCE {
      isMrIndex                    INTEGER,
      isMrUserName                 DisplayString,
      isMrLastTime                 DateAndTime,
      isMrLastDocument     DisplayString
      }
```

```
isMrIndex OBJECT-TYPE
      SYNTAX          INTEGER (0..128)
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The index indicates the sequence of the recent users."
      ::= { isMostRecentNUserEntry 1 }
      -- Instrumentation : Agent internal

isMrUserName OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The name of the user."
      ::= { isMostRecentNUserEntry 2 }
      -- Instrumentation : Log file

isMrLastTime OBJECT-TYPE
      SYNTAX          DateAndTime
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The last time the user has accessed this server."
      ::= { isMostRecentNUserEntry 3 }
      -- Instrumentation : Log file

isMrLastDocument OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The last operation or document which is accessed
                  by the user at this server."
      ::= { isMostRecentNUserEntry 4 }
      -- Instrumentation : Log file

--
-- The section of the errors that occured in the Information Store.
--

isErrors OBJECT IDENTIFIER ::= { isObjects 3 }
```

```
isErrorTable OBJECT-TYPE
      SYNTAX          SEQUENCE OF IsErrorEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The access of document table keeps track of the
                  number of accesses that have been done in the past.
                  Each entry is a document.

                  This is a Top N table with the N most recently accessed
                  documents."
      ::= { isErrors 12 }

isErrorEntry OBJECT-TYPE
      SYNTAX          IsErrorEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "Entry (conceptual row) of the document table."
      INDEX       { errorIndex }
      ::= { isErrorTable 1 }

IsErrorEntry ::= SEQUENCE {
      isErrorIndex                 INTEGER,
      isErrorDescription           DisplayString,
      isErrorCount                 Counter32,
      isErrorLastTime              TimeStamp
      }

isErrorIndex OBJECT-TYPE
      SYNTAX               INTEGER (0..128)
      MAX-ACCESS  not-accessible
      STATUS               current
      DESCRIPTION
                  "The index indicating the error."
      ::= { isErrorEntry 1 }

isErrorDescription OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The textual description of the error."
      ::= { isErrorEntry 2 }

isErrorCount OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The number of errors that occured."
      ::= { isErrorEntry 3 }
```

```
isErrorLastTime OBJECT-TYPE
      SYNTAX          TimeStamp
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The time of the last error occured."
      ::= { isErrorEntry 4 }

--
-- This group contains information about documents which are accessed
-- on the Information Store Server. That are the documents accessed since
-- the start of the server. This are the URI with which a information
-- can be accessed.

-- For example a document is index.html:
--    If this is a document retrieved from the directory root of the
--                Information Store.
--    If this document is in a directory beneath the directory root it
--                should also mention this path in the isDocumentName.
--                <path>'/'<name of script>
--
-- For example a document script of the cgi-interface:
--    The script is mentioned in this table with the isDocumentName
--                ‘cgi/'<path>'/'<name of script>
--

isDocuments OBJECT IDENTIFIER ::= { isObjects 4 }

isDocumentTableSize OBJECT-TYPE
      SYNTAX          INTEGER (0..128)
      MAX-ACCESS  read-write
      STATUS          current
      DESCRIPTION
                  "The number of entries which are contained
                  in the isDocumentTable."
      ::= { isDocuments 1 }

isDocumentTableRefresh OBJECT-TYPE
      SYNTAX          TruthValue
      MAX-ACCESS  read-write
      STATUS      current
      DESCRIPTION
                  "Field with which a new ‘document table' will be made."
      ::= { isDocuments 2 }

isDocumentTableDate OBJECT-TYPE
      SYNTAX          DateAndTime
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The universal date and time for the last time the
                  isDocumentTable was refreshed."
      ::= { isDocuments 3 }
```

```
isDocumentTable OBJECT-TYPE
      SYNTAX          SEQUENCE OF IsDocumentEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The access of document table keeps track of the
                  number of accesses that have been done in the past.
                  Each entry is a document."
      ::= { isDocuments 4 }

isDocumentEntry OBJECT-TYPE
      SYNTAX          IsDocumentEntry
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "Entry (conceptual row) of the document table."
      INDEX { documentIndex }
      ::= { isDocumentTable 1 }

IsDocumentEntry ::= SEQUENCE {
      isDocumentIndex              INTEGER,
      isDocumentName               DisplayString,
      isDocumentAccesses   Counter32,
      isDocumentAccessRightAccessRights,
      isDocumentSize               Integer32,
      isDocumentErrors             Counter32,
      isDocumentUpdate             DateAndTime,
      isDocumentAccessesAtLastUpdateCounter32,
      isDocumentLastAccess DateAndTime,
      isDocumentType               MediaType
      }

isDocumentIndex OBJECT-TYPE
      SYNTAX          INTEGER (0..127)
      MAX-ACCESS  not-accessible
      STATUS          current
      DESCRIPTION
                  "The ranking of the document is a top N table."
      ::= { isDocumentEntry 1 }
      -- Instrumentation : Log file

isDocumentName OBJECT-TYPE
      SYNTAX          DisplayString
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The name of the document available at the Information
                Store."
      ::= { isDocumentEntry 2 }
      -- Instrumentation : Log file
```

```
isDocumentAccesses OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The number of accesses of the document."
      ::= { isDocumentEntry 3 }
      -- Instrumentation : Log file

isDocumentAccessRights OBJECT-TYPE
      SYNTAX          AccessRights
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The accessrights of the document. This field should
                  restrict the information retrieval of the clients."
      ::= { isDocumentEntry 4 }
      -- Instrumentation : System

isDocumentSize OBJECT-TYPE
      SYNTAX          Integer32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The size of the document in bytes (octets)."
      ::= { isDocumentEntry 5 }
      -- Instrumentation : System

isDocumentErrors OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The number of errors that occured during accessing
                  this document."
      ::= { isDocumentEntry 6 }
      -- Instrumentation : Log file

isDocumentUpdate OBJECT-TYPE
      SYNTAX          DateAndTime
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "Date and time when this file was last updated."
      ::= { isDocumentEntry 7 }
      -- Instrumentation : System
```

```
isDocumentAccessesAtLastUpdate OBJECT-TYPE
      SYNTAX          Counter32
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The number of accesses of the document since
                  the document was updated for the last time."
      ::= { isDocumentEntry 8 }
      -- Instrumentation : System

isDocumentLastAccess OBJECT-TYPE
      SYNTAX          DateAndTime
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The date and time the document was last accessed."
      ::= { isDocumentEntry 9 }
      -- Instrumentation : Log file

isDocumentType OBJECT-TYPE
      SYNTAX          MediaType
      MAX-ACCESS  read-only
      STATUS          current
      DESCRIPTION
                  "The type of the document."
      ::= { isDocumentEntry 10 }
      -- Instrumentation : -

--
-- Conformance and compliance definitions.
--

isGeneralGroup OBJECT-GROUP
      OBJECTS {   isName,
                  isOrganisation,
                  isContact,
                  isLastInitialisation,
                  isSupportedMediaTypes,
                  isTopic }
      STATUS      current
      DESCRIPTION
                  "The isGeneralGroup defines the objects
                  which are common to all Information Store
                  applications"
      ::= { isGroups 1 }
```

```
isApplDependancyGroup OBJECT-GROUP
      OBJECTS{
        isApplResourceIdentifier, isApplProcessName,
                  isApplVersion, isApplUptime,
                  isApplOperStatus, isApplLastChange,
                  isApplLastActivity, isApplFailedActivities }
      STATUS      current
      DESCRIPTION
                  "The application dependency Group provides information
                  about additional processes (or helper processes) of
                  the information store."
      ::= { isGroups 2 }

isErrorGroup OBJECT-GROUP
      OBJECTS {   isErrorIndex, isErrorDescription,
                  isErrorCount, isErrorLastTime }
      STATUS      current
      DESCRIPTION
                  "The isErrorGroup provides the information
                  about the errors occured with the
                  information store."
      ::= { isGroups 3 }

isAccessGroup OBJECT-GROUP
      OBJECTS {
                  isNumberOfAccesses,
                  isNumberOfBytesOut,
                  isNumberOfBytesIn,
                  isDomainName, isDomainAccesses,
                  isAccessOfLastNDaysTableSize,
                  isLastNDayIndex, isAccessOflastNDay,
                  isMrUserName, isMrLastTime, isMrLastDocument,
                  isMostRecentNUserTableSize,
                  isMostFrequentNUserTableSize,
                  isMostFrequentNUserTableRefresh,
                  isMostFrequentNUserTableDate,
                  isMfUserName, isMfNumberOfAccesses }
      STATUS      current
      DESCRIPTION
                  "The isAccessGroup provides the information
                  about the accesses performed on this
                  information store..
                  The group defines the total number of accesses
                  and the number of bytes belonging to the accesses.
                  The otherwise the accesses or ordered in tables
                  The ordination are:
                          by domain,
                          by day (LastNDays),
                          by most frequent users (TopN),
                          by most recent users (LastN)"
      ::= { isGroups 4 }
```

```
isDocumentGroup OBJECT-GROUP
      OBJECTS {
                  isDocumentTableSize,
                  isDocumentTableRefresh,
                  isDocumentTableDate,
                  isDocumentName,
                isDocumentAccesses, isDocumentAccessRights,
                  isDocumentSize, isDocumentErrors, isDocumentUpdate,
                  isDocumentAccessesAtLastUpdate, isDocumentLastAccess,
                  isDocumentType }
      STATUS      current
      DESCRIPTION
                  "The isDocumentGroup provides the information
                  about the accesses which can be made on the
                  information store."
      ::= { isGroups 5 }

isMIBCompliance MODULE-COMPLIANCE
      STATUS      current
      DESCRIPTION
                  "The compliance statements for http entities
                  which implement the HTTP MIB"
      MODULE
                  MANDATORY-GROUPS {
                                    isGeneralGroup,
                                    isApplDependancyGroup,
                                    isErrorGroup,
                                    isAccessGroup,
                                    isDocumentGroup }
      ::= { isCompliances 1 }

END
```

6. Acknowledgments

This document has been produced by the University of Twente (The Netherlands), together with ESYS Limited (The United Kingdom), as part of a ‘proof of concept' study for the ‘Centre of Earth Observation' (CEO) of the ‘Joint Research Centre' (JRC) of the European Community. This document has benefited greatly to the comments of:

Mark Gamble
<mgamble@esys1.esys.co.uk>

Rui Meneses
<rui.meneses@jrc.it>

Juergen Schoenwaelder
<schoenw@cs.utwente.nl>

7. References

[1] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1902, January 1996.

[2] McCloghrie, K., and M. Rose, Editors, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", STD 17, RFC 1213, Hughes LAN Systems, Performance Systems International, March 1991.

[3] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Textual Conventions for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1903, January 1996.

[4] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1905, January 1996.

[5] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Conformance Statements for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1904, January 1996.

[6] Case, J., M. Fedor, M. Schoffstall, J. Davin, "Simple Network Management Protocol", RFC 1157, SNMP Research, Performance Systems International, MIT Laboratory for Computer Science, May 1990.

[7] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Introduction to Community-based SNMPv2", RFC 1901, January 1996.

8. Security Considerations

Security issues are not discussed in this memo.

9. Authors' Addresses

Harrie Hazewinkel / Eric van Hengstum / Aiko Pras
University of Twente
Centre for Telematics and Information Technology (CTIT)
POBox 217
7500 AE Enschede, The Netherlands
Phone: +31-53-4893778
Email: hazewink@cs.utwente.nl
       hengstum@cs.utwente.nl
       pras@cs.utwente.nl

Table of Contents