SNMP – MIB Browser

Laia Valentí i Zurriaga

Academic Supervisor: Josep Paradells Aspas
GNMS Team Leader: Baard Grindberg
GNMS Leader: Baard Grindberg

Project done at British Telecom UK, Martlesham Heath, Ipswich, UK

BTexact Technologies www.btexact.com Adastral Park, Martlesham, Ipswich, Suffolk IP5 3RE, UK

RESUM

En primer lloc, voldria explicar que aquest projecte va ser realitzat l'any 2004, però per un seguit de motius no ha estat presentat a l'escola fins el dia d'avui. Tot i així, tots el textos han estat revisats recentment i les tecnologies i aplicacions emprades són vigents en l'actualitat.

Aquest projecte va ser realitzat en el centre d'I+D de l'empresa British Telecom ubicat a Ipswich (Anglaterra), en el departament Broadcast and Satellite Communication (B&SC).

L'objectiu del treball fou desenvolupar una nova aplicació (MIB Browser) per administrar xarxes i components de sistema utilitzant el protocol SNMP (Simple Network Management System). Aquesta aplicació va ser integrada en el sistema de control i gestió de xarxa GNMS (Global Network Management System) que estava desenvolupant el departament del qual vaig començar a formar part.

El MIB Browser consisteix en una aplicació que s'utilitza per consultar i configurar elements dels dispositius d'una xarxa. Aquests elements, anomenats MIB variables, contenen informació de la xarxa i estan definits en els MIB modules. Els MIB Modules són fitxers plans que contenen les definicions i estructura d'aquestes variables consultables i configurables.

El primer pas va ser dissenyar una estructura eficaç per emmagatzemar les MIB variables en la base de dades relacional utilitzada en el GNMS, per tal d'aconseguir una forma ràpida d'accedir i treballar amb la informació.

Un cop creada la base de dades, es va desenvolupar una aplicació web, per carregar i eliminar els MIB Modules i les seves variables en la base de dades.

Finalment es va implementar el MIB Browser, es tracta d'una aplicació web en la que es pot explorar de forma ràpida totes les variables emmagatzemades en la base de dades, permetent a l'usuari consultar i configurar l'estat d'aquestes en els equips de la xarxa.

AGRAÏMENTS

M’agradaria dedicar el projecte als meus pares, Maria Dolors i Josep, gràcies a ells he pogut dur a terme els estudis d’enginyeria i en conseqüència la realització d’aquest.

Vull fer una especial menció a dues persones que m’han ajudat en el desenvolupament del projecte, el meu cap de projecte Baard Grindberg, que em va donar llibertat per investigar i recercar, sense deixar de donar-me consells i compartint els seus coneixements amb mi. I al meu tutor Josep Paradells, per recolzar-me quan, passats 4 anys, vaig decidir presentar el projecte, i per ajudar-me activament amb l’estructura i continguts.

També voldria reconèixer el gran suport i recolzament que vaig rebre del meu amic Jordi.

Finalment, m’agradaria donar el meu agraïment a tots els amics que em van fer més fàcil estar lluny del meu país; als nous amics d’Anglaterra, que em feien sentir com a casa, especialment el Pedro, el Pere, l’Eduardo i el Roberto. Agrair també als amics de Barcelona i la meva germana Mireia l’haver-me fet sentir a prop dels meus.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 OVERVIEW

This project was developed in *British Telecom* (BT) UK at Martlesham Heath (Ipswich, Suffolk) as a part of an internship in agreement with the Universitat Politècnica de Catalunya (UPC).

During my placement I joined an active group in the *Broadcast and Satellite Communications* (B&SC's) department contributing in the development of new application and solution for GNMS (Global Network Management System), GNMS was being developed to rationalize B&SC's existing network management systems. The objective of my project was to develop a new MIB BROWSER for administering network and system components using the SNMP (Simple Network Management System) protocol.

The following sections give a background summary on GNMS and identify where the MIB BROWSER fits in.

1.1 GLOBAL NETWORK MANAGEMENT SYSTEM (GNMS)

BT Ignite, Broadcast and Satellite Communications has a large number of management systems deployed to handle their Terrestrial, IP and Satellite broadcast networks. The current systems have been developed over a number of years to meet individual business requirements as and when those requirements emerged. The majority of these systems were developed in short time scales and integrated vertically, whereby horizontal integration is practically non-existent.

Data is spread widely across a large number of databases and platforms, which has resulted in significant duplication of data, leading to fragmentation and inaccuracy. There is also limited automated electronic flow of information between the Network management systems and the scheduling and billing systems.

In summary, the existing systems meet the current service delivery requirements, but are inflexible and inconsistent. They are expensive to operate, difficult to maintain and do not fully support the key business objectives.

The Global Network Management System is the next system that will be used to manage the B&SC broadcast networks. GNMS will replace several old proprietary management systems and integrate with others still in use. In other words, GNMS development is part of a program to rationalize the existing Network Management Systems.

Its objective is to group systems with similar characteristics into a reduced set of new web-based applications. This system shall be implemented under a

unified standard web interface and shall contain the features of a number of existing systems.

GNMS Architecture promotes:

- *System Expandability*: GNMS is designed using an object-oriented approach, where functionality is encapsulated in a set of processing modules. These modules are all accessed using standard interfaces. The design creates a system core that implements the functionality of the system, over which new modules can be added as required.

- *Cost-effective System Maintenance*: Object-oriented design approach eases system maintenance, as faults and enhancements may be isolated to a particular module. Hence, updates may be done with minimum impact on other modules and their functionality.

- *Component Reuse:* Each of the modules is an autonomous entity providing a unique set of functions. These functions will be available to external systems through pre-defined interfaces.

- *Reduced Development Time Scales*: Once the module interfaces have been defined, each of the modules can be implemented in parallel. With sufficient resources we can design and develop in shorter time scales compared to non-modular systems. The modules can also be out-sourced to external suppliers, resulting in shorter time scales overall.

1.2 MIB BROWSER

BTExact's Operational Support System (OSS) has been using a MIB Browser application since the 2001. The term MIB Browser refers to a class of generic management applications that are used to query network devices to read and set the values of its MIB (*Management Information Base*) variables, these variables hold network information, examples of it include the system description, the IP address assigned to an interface and the number of incoming bytes received at an interface. A set of MIB variables are defined in MIB Modules

The aim of the project was to replace the old MIB Browser for a new one that used the Database Services provided by the GNMS Core Modules achieving in that way more efficiency.

The first objective was to find a suitable storage structure for MIB Modules in a relational database that would allow for fast retrieval of the management information. This database was integrated into the GNMS database (GNdB).

The second was to develop a web application (MIB Manager) to store the MIB variables loading the MIB modules into the database and deleting them from it.

The last one was to create the MIB Browser, it consists in a web-based application integrated in the GNC module, in GNC the MIB browser will allow a user to explore the MIB structure, to query the equipments for status, and display its configuration for possible modification.

1.3 ORGANIZATION OF THE THESIS

The project is organized as follows:

Chapter 1 gives an explanation of the Global Network Management System structure and introduces the SNMP MIB Browser application.

Chapter 2 explains what is network management and why is needed.

Chapter 3 is used to describe the main theoretical part of the report, the SNMP

Chapter 4 describes the System Architecture of the SNMP Mib Browser application, giving a brief explanation of the main parts of the project and talking about the requirements that it must satisfy.

Chapter 5 gives an explanation of the background elements used in the application, such as libraries, software tools, database connection, XML, XSLT…

Chapter 6 presents the MIB Browser, giving a complete description of it.

Chapter 7, and 8 are MIB Manager and MIB Browser user guides respectively.

Chapter 9 contains the application validation.

Chapter 10 contains the conclusions.

2. NETWORK MANAGEMENT SYSTEM

This chapter describes functions common to most network-management architectures and protocols. It explains why is necessary to manage a network in the current time and it also presents the five conceptual areas of management as defined by the International Organization for Standardization (ISO).

2.1 WHAT IS NETWORK MANAGEMENT SYSTEM?

Network management is the process of controlling a complex data network so as to maximize its efficiency and productivity. In their discussion on the basics of network management, Cisco [1] Systems point out that the term "network management" means different things to different people. They give two examples at opposite ends of the spectrum to illustrate this diversity: A solitary network consultant monitoring network activity and high end workstations generating graphical views of network topologies and traffic. In general, network management is a service that employs a variety of tools, applications, and devices to assist human network managers in monitoring and maintaining networks.

There are several organizations who have developed services, protocols and architectures for network management. The three most important organizations are:

- The International Organization for Standardization (ISO).
- The Comité Consultative Internationale de Telegraphique et Telephonique (CCITT); this organization is nowadays called the Telecommunication Standardization Sector (ITU-T) of the International Telecommunication Union (ITU).
- The Internet Engineering Task Force (IETF).

Of these three ISO was the first who started, as part of its “Open Systems Interconnection” (OSI) program, the development of an architecture for network management. The first proposals for such an architecture appeared during the early 1980; nowadays a large number of standards exist for the architecture as well as for network management services and protocols. Of these standards the “OSI Management Framework”, the “OSI Systems Management Overview” and the “Common Management Information Protocol” (CMIP) are probably the best known examples.

Initially the aim of ISO was to define management standards for datacom networks; development of management standards for telecom networks was left to CCITT. In 1985 CCITT started the development of such management standards; these standards have become known as the “Telecommunications Management Network” (TMN) recommendations. Originally these

[1] *Network Management Basics*, Cisco Systems, February 2002.

recommendations were self standing, but during the 1988-1992 study period they have been rewritten to include the ideas of OSI management. Nowadays OSI management and TMN can be seen as each others complements.

Looking back at the last decade it may be concluded that the growth of the Internet has played a decisive role in the development of network management protocols. Initially the Internet Architecture Board (IAB) intended to apply the OSI management approach, but at the time the size of the Internet reached a level at which management became indispensable, OSI management groups were still busy with discussing the OSI management framework. Since implementations of OSI management were not expected to appear soon, the IAB requested the IETF (the organization who is responsible for the development of Internet protocols) to define an ad hoc management protocol. This “Simple Network Management Protocol” (SNMP) was completed within a year and soon many manufacturers started the production of SNMP compliant systems.
Although SNMP has several deficiencies, it has become the de facto standard for management of datacom networks. In 1993 an attempt was made to tackle these deficiencies and an improved version of SNMP (SNMPv2) appeared. The last version of SNMP is SNMPv3 and is the one that is used nowadays.

2.1.1 OSI Management

The International Telecommunications Union (ITU) proposed a network management model aimed at understanding the major functions of network management and monitoring software.
This management model forms part of the X.700 series of documents from the ITU [2] and is based on the Open Systems Interconnect (OSI) reference model. It has been standardised by the International Standards Organisation (ISO). The ISO has contributed a great deal to network standardization. Its network management model is the primary means for understanding the major functions of network management systems.

2.1.1.1 Functional areas

The model consists of five conceptual areas [3]:

- **Fault Management**: Detect, isolate, notify, and correct faults encountered in the network.
- **Configuration Management**: Configuration aspects of network devices such as configuration file management, inventory management, and software management.

[2] SO 7498: "Information Processing Systems - Open Systems Interconnection -Basic Reference Model", Geneva, 1984

[3] *Network Management System: Best Practices White Paper*, Cisco Systems, August 2004

- **Performance Management**: Monitor and measure various aspects of performance so that overall performance can be maintained at an acceptable level.
- **Security Management**: Provide access to network devices and corporate resources to authorized individuals.
- **Accounting Management**: Usage information of network resources.

Fault Management

Fault management is the process of identifying and correcting network problems, otherwise known as faults. Faults typically manifest themselves as transmission errors or failures in the equipment or interface. Faults result in unexpected downtime, performance degradation and loss of data. Generally, fault conditions need to be resolved as quickly as possible.

Comprehensive fault management is the most important task in network management. Fault management tools can help increase the reliability of the network by quickly identifying the fault, and then help initiate the recovery process.

The first step is to identify the fault, isolate the cause of the fault, and then, if possible, correct the fault. This three step process requires predetermining which faults should be managed and be given higher priorities than others, and then utilizing a set of tools to resolve the fault through the network manager. In the simplest case, an alarm is detected, and a maintenance technician is dispatched to locate and resolve the fault at the suspect location.

Using more advanced tools, the network manager may be able to go many steps further to isolate and correct a fault from the manager location, and then return the network to normal operation without the user being aware of the failure.

Configuration Management

Configuration management deals with the initialization, modification, and shutdown of a network. Networks are continually adjusted when devices are added, removed, reconfigured, or updated. These changes may be intentional, such as adding a new server to the network, or path related, such as a fiber cut between two devices resulting in a rerouted path. If a network is to be turned off, then a graceful shutdown in a prescribed sequence is performed as part of the configuration management process. The process of configuration management involves identifying the network components and their connections, collecting each device's configuration information, and defining the relationship between network components. In order to perform these tasks, the network manager needs topological information about the network, device configuration information, and control of the network component.

Performance Management

Performance management involves measuring the performance of a network and its resources in terms of utilization, throughput, error rates, and response times. With performance management information, a network manager can reduce or prevent network overcrowding and inaccessibility. This helps provide a more consistent level of service to users on the network, without overtaxing the capacity of devices and links. This form of management looks at the percentage of utilization of devices and error rates to help in improving and balancing the throughput of traffic in all parts of a network. Typically, some devices are more highly utilized than others. Performance monitoring give qualitative and time relevant information on the health and performance of devices so that underutilized devices are more fully utilized, and overtaxed devices are rebalanced. In a well-utilized network with healthy components, the error rates for packets traversing the network are down and response times are shortened.

Security Management

Security management deals with ensuring overall security of the network, including protecting sensitive information through the control of access points to that information. Sensitive information is any data that an organization wants to secure, such as research documents, payroll data, and sales and inventory figures. Protecting sensitive data from unauthorized access is a common requirement. Security concerns can be assuaged with a well-designed and implemented security management system.

Security management controls access to the network devices and sensitive information through the use of devices such as passwords. This management also controls the form of sensitive data using methods such as encryption. There are many encryption techniques available for sensitive digital data such as public and private key encryption that have been in use for some time.

Each area of management in this five-part model is not exclusive of the others. It is typical for performance management to work in conjunction with fault and configuration management. For example, a monitored device interface that exhibits a slowly increasing error rate can be verified by an alarm query, and could then be reconfigured and bypassed before the fault affects traffic. Using these five management categories as guide lines, a more critical review can be made of any product's ability to manage or be managed as a network device.

Accounting Management

Accounting management involves tracking each individual user's utilization of network resources for the purposes of allocation of resources and billing for their use of the network. This type of information helps a network manager allocate the right kind of resources to users, as well as plan for network growth. With the same information, the cost of transmitting messages across the network can be computed and billed to the user if the traffic was revenue bearing. This type of management involves monitoring the login and logoff

records, and checking the network usage to determine a user's use of the network. In addition, access privileges and usage quotas can be established and checked against actual for accounting information.

2.1.1.2 Managed objects, management information and the MIB

To understand the relationship between managed objects, management information and the 'Management Information Base' (MIB), it may be helpful to take a look at the development of the Management Framework standard. Several draft versions of this standard contained the following definitions:

- *Managed object:* "those data processing and data communications resources (whether OSI resources or not) that may be managed through the use of an OSI management protocol".
- *Management information:* "Information *associated* with a managed object that is operated on by the OSI Management protocol to control and monitor that object".

These definitions suggest that a difference exists between managed objects and management information. Although the various drafts of the Management Framework document are sometimes difficult to understand, also the following view emerges:

- managed objects reside in the various layers of the OSI RM,
- management information resides in the *Management Information Base* (MIB).

The MIB can be seen as a kind of database. The contents of this database are not the set of managed objects themselves, but the information that is *associated* with the managed objects. Layer Managers (LMs) are responsible to maintain the association between MIB information and managed objects (Figure 2.1). In case of problems with Layer Managers, it might occur that the information in the MIB does not accurately reflect the state of the managed objects any more.

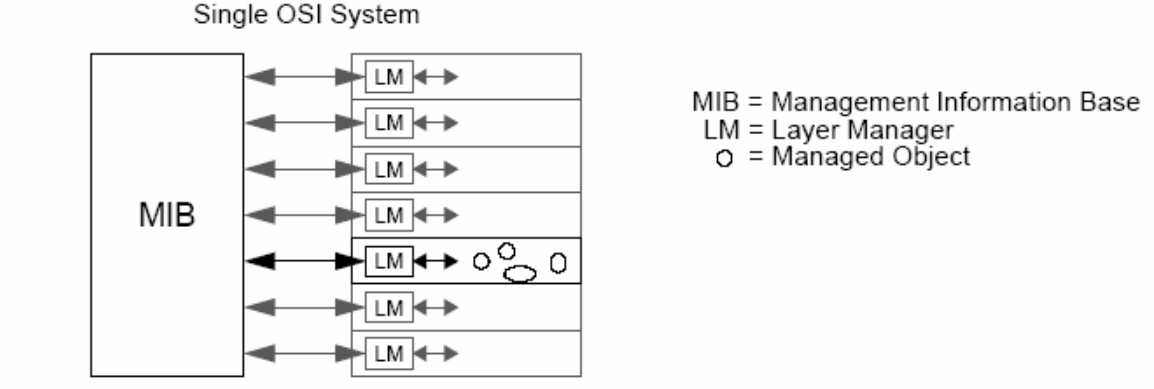

Figure 2.1 MIB, Managed Objects and Layers Managers

2.1.2 TMN Management

The TMN Management information follows the patterns set in the ISO model but they are expanded to offer a catalogue of objects classes adapted to the specific modelling needs found in the telecommunication network elements[4]. Particularly relevant in this structure of management information are those objects classes related to the topological specification of the network. On the other hand, the standardized protocols, specified as a part of the Q.3 interface[5], use the layered ISO structure where the CMIP plays the main role as the management protocol.

The functions in the TMN follow, in some way, the organization in five basic functional areas defined in the ISO model. But, in the other hand, each one of these functions is structured in four layers: element management, network management, service management and business management.

2.1.3 Internet Management

The Internet model for managing information is based on a very simple set of protocols and structures of information. The structure of information follows the object-oriented model but introducing many simplifications with regard to the TMN model. This situation makes the modelling of complex devices, networks and services more difficult but, on the other hand, it makes the management applications considerably easier. While an Internet MIB can always be described according to the TMN model, the opposite is not true. Many features modelling objects in TMN have no translation to the Internet management model.

On the other hand, the network management protocol follows the layered structure of Internet (more limited than the OSI model of TMN) using the SNMP (Simple Network Management Protocol) as the management protocol, usually on top of a stack of TCP/IP protocols. The SNMP protocol, designed with the aim of simplicity, is able to manage many features of many devices, but it has severe limitations for managing complex objects or a structure of objects.

The limitations of the Internet model are, at the same time, an advantage and a drawback. On the one hand it is clear that it limits the capabilities of managing the network. But, on the other hand, its easy design permits an easy development of agents and managers, functions and management platforms. This situation has led to a fast commercial expansion. There are many devices supporting the SNMP management protocol, while only the high-end systems support the CMIP standard. Additionally, many vendors provide management platforms based on SNMP, with similar functions and performances and at very attractive prices. The greater simplicity of the

[4] International Telecommunication Union- Telecommunications Sector. ITU-T Rec. M.3100.*Generic Managed Object Class Library.* ITU-T, 1995

[5] International Telecommunication Union- Telecommunications Sector. ITU-T Rec. Q.812. *Upper layer protocols profiles for the Q.3 interface.* ITU-T, 1993

Internet model, its greater market share, and the greater number of competitors make a situation of lower prices possible.

2.2 NEED FOR NETWORK MANAGEMENT

Networks are of growing importance and have become critical in the business world. Networks are getting more and more complex and heterogeneous, i.e. different types of networks (computer networks, telephone networks, mobile cellular networks and others) are working together. It is not just the public networks that grow and gets more complex, but also within organizations.

Organizations want to take advantage of the technology and they build complex networks. As the networks become larger and more complex and heterogeneous, the costs rise. In spite of the networks being complex, the demands are still very high, they should or must work 24 hours a day. It is a big challenge to have all services running and offer good quality of service (QoS). This is why network management systems are needed. They discover faults and errors in networks and sometimes correct them, they discover performance issues, they secure the network, they ease the configuration of the network and more. This makes one able to catch network failures before they are too critical. In short, a network management system monitors the health of the network. If something goes wrong, network engineers are alerted and the problem is hopefully fixed before the network users notice any problems.

2.2.1 Why is there Network Management?

There are several types of networks, some of them transport data. The two main networks that can transport data are the computer networks and telecommunication networks. The telecom networks are very reliable. You can call anyone, anytime, from anywhere to anywhere in the world and be almost sure that you get connected to the destination. The computer networks are not equally reliable because of several things. Computer technology is more complicated than telephone services. Computer communication is mostly packet switched and there is therefore no guarantee that you get a certain amount of bandwidth. The telecommunications network, however, use circuit switched messages and therefore guarantees a certain amount of bandwidth. Further, the telephone industry all over the world has been monopolistic and single-vendor. This is not the case anymore and the telecommunication networks are also multi-vendor and heterogeneous with complexity increasing because of new services are being developed. Computer networks were multi-vendor from the start. The computer technology also has more rapid standardization processes and development cycles compared to the telecommunication (where ITU-T is the standardization organization).

Figure 2.2 illustrates the complexity of computer technology and as the computer networks, telecommunication networks and also mobile cellular networks are merging, the complexity grows even more. For example, one can browse the Web using the telephone wires and one can dial someone from a computer.

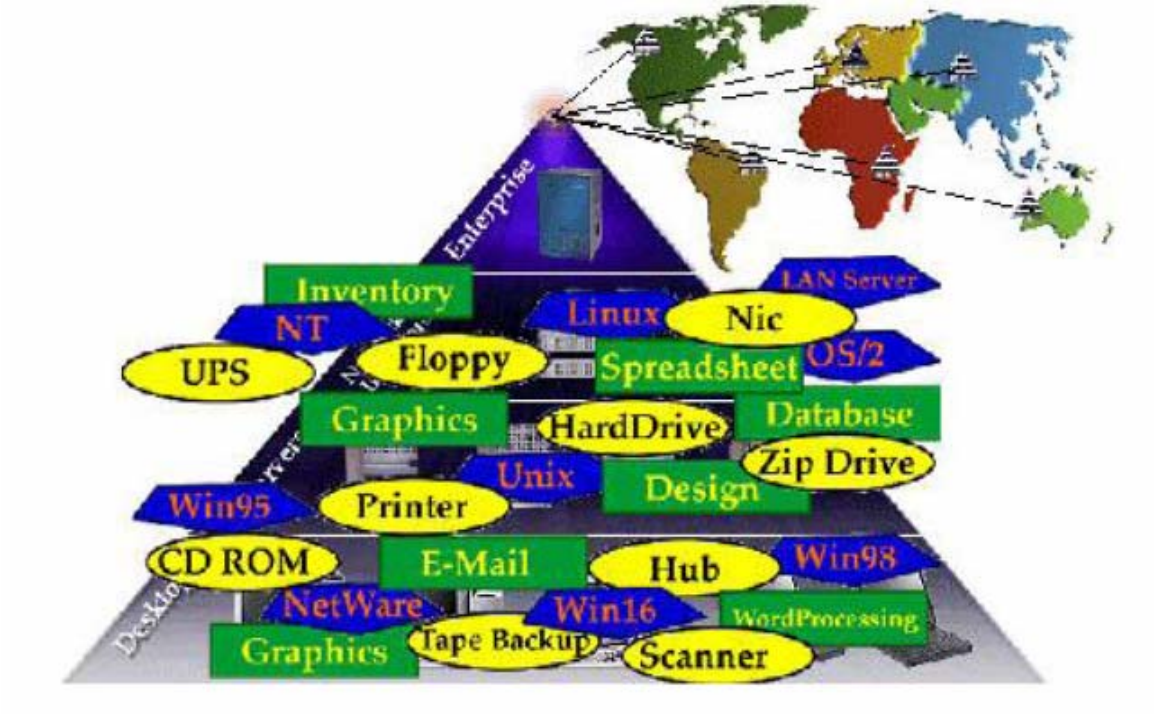

Figure 2.2 Complexity due to that everything should work together.

A network can be very complex and it must often perform well and be secure. One solution to control the network is to do these checks manually. For example, one can ping the network devices to get their respond times. One can manually scan the network for security vulnerabilities. This will be tedious and error prone work, especially in a large network. Maintaining a network includes very many tasks and performance and security checks must be made regularly. This is why network management is needed. Introducing a network management system to a network automates most of the network management tasks and makes other tasks easier to perform. If there was only one network, for instance the computer network, we would still need network management. But due to the complexity and that people have high demands (everything should always work), network management gets more and more important.

2.2.2 Who Needs Network Management?

Almost any enterprise makes use of an internal network. If a company connects its computers together in a network environment, then a management system is probably needed. When a network fails, or shows poor performance, the costs can be enormous, productivity of employees can suffer, dissatisfaction among users and customers could cause other

problems. If cost of ownership, reliability, performance and availability matter, an organization will probably need a network management system.

In conclusion, the goals of the network management are:

- Networks become larger and more complicated
- To keep system with high performance and availability
- Provide immediate assistance with problems

2.3 NETWORK MANAGEMENT ARCHITECTURE

Most network management architectures use the same basic structure and set of relationships.

The architecture consists of the following elements (see figure 2.3):

- Management station
- Management agent
- Management information base
- Network management protocol

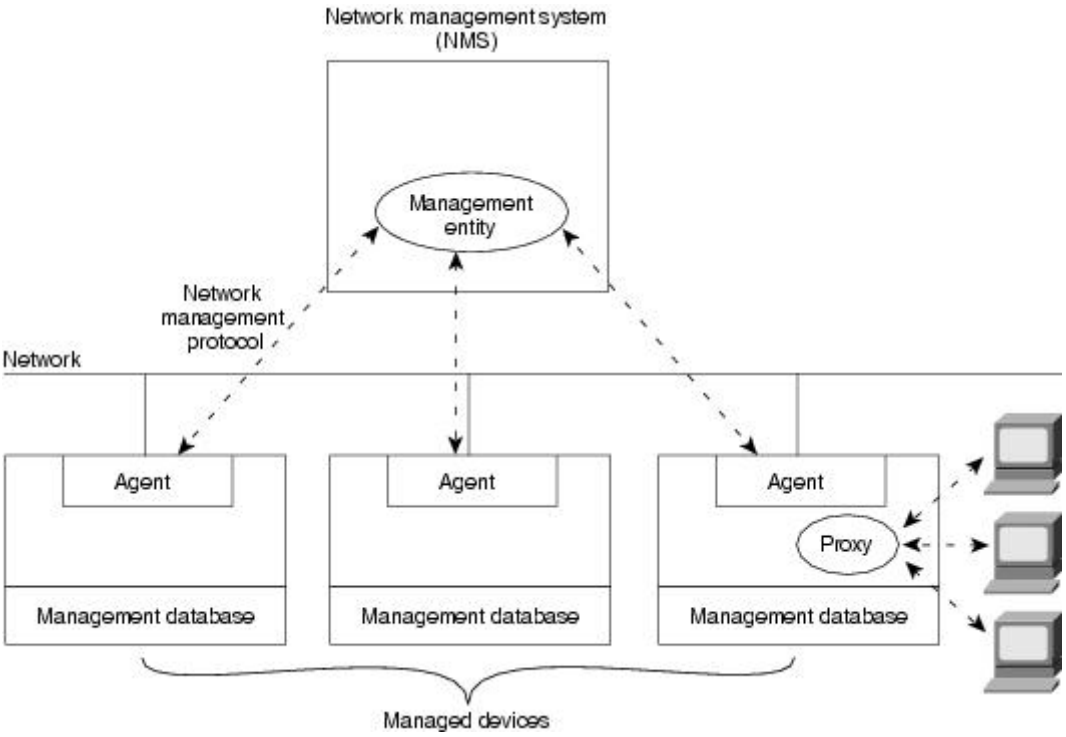

Figure 2.3 Network Management Architecture

A **management station** provides the interface for a human network manager to interact with the management system. This can be a workstation or a PC that executes a set of application programs for data analysis, fault management, data presentation, etc. The station is capable of translating the network manager's requirements into the actual tasks of monitoring and controlling individual network devices. In addition, it contains the database of information extracted from all the managed entities in the network.

There are many popular network management platforms on the market standards and cannot interoperate with any network device other than their own products. This is a problem for network managers because the network may include equipment from different vendors that do not subscribe to the same proprietary platform. These devices cannot be managed by the management system; the manager is "blind" to these devices' activity and performance.

The **management agent** is the second active element in the management architecture. The agent is a software program in the network device that responds to requests for information or actions issued by the management station. The agent may also send the station unsolicited information, known as a Trap. Such a program is very much tied to the internal workings of the device. All devices in a network must have a management agent. Typically, an agent may be embedded or "native" to the device, or alternatively be a "proxy" agent for other protocols.
The third part of the architecture is the **information** that is exchanged between the manager and the agent. This information is a collection of objects or data values, each represents one aspect of the managed device.

The last element of the model, the network management protocol, links the station and the agent by specifying the rules for communication. The two most common network management protocols are the:

- Simple network Management Protocol
- Common Management Information Protocol

SNMP is by far the most widely used network management protocol and use is widespread in LAN environments. CMIP is used extensively in telecommunication environments, where networks tend to be large and complex.

3. SNMP (SIMPLE NETWORK MANAGEMENT SYSTEM)

3.1 INTRODUCTION

Before explaining where the SNMP comes from it would be necessary to introduce briefly how protocols are found that meet the requirements of the research and commercial communities, who determines that a protocol solution is adequate and who chooses between competing solutions.

The Internet Architecture Board (IAB)

The IAB oversees the Internet protocol development process, focussing effort on areas that need work and deciding if and when a protocol is ready to be admitted as an Internet standard.

The Internet Engineering Task Force (IETF)

The IAB sets general directions and decides which protocols will be standards. The work of sifting through requirements and designing, implementing, and testing new standards is carried out by the IETF. The IETF is a loosely defined group of people with networking know-how: researchers, designers, people with operating experience, and equipment vendor engineers.

The Internet Engineering Steering Group (IESG)

The *Internet Engineering Steering Group* (IESG) is made up of IETF chair people and others performing a leadership role in the IETF. The IESG performs an oversight and coordinating function for the IETF.

The Internet Research Task Force (IRTF)

Long-term architectural issues are considered by another group called the *Internet Research Task Force* (IRTF). Its steering group is the *Internet Research Steering Group* (IRSG).

Requests For Comments and Standards

Internet standards are published within a series documents called Request For Comments (RFCs). There are RFCs on many topics, ranging from how to name your computer to experimental results in networking research.

An RFC under serious consideration for incorporation in the Internet protocol suite will be published as a proposed standard. If all goes well, it can be promoted to draft standard after at least six month. At this point there must be two or more implementations of the protocol. Finally, after a review period of at least four more months, if the IESG recommends adoption as a standard, the IAB makes the final decision on adoption.

The standards development process is an extremely open one. RFC documents are freely available to anyone, and anyone can comment on these documents at any stage.

The standards track process is illustrated in Figure 3.1. Often a protocol will need to be modified substantially during this process. In this case, a new RFC will be published detailing the corrected protocol and obsoleting the old protocol.

A BRIEF OVERVIEW OF TCP/IP

SNMP has management of TCP/IP networks as its first objective. We'll present a brief overview of the TCP/IP protocol suite here.

The ISO model

The International Organization for Standardization (ISO) Open Systems Interconnect (OSI) model is a useful blueprint for identifying the major components of TCP/IP. Figure 3.1 compares TCP/IP and OSI layers.

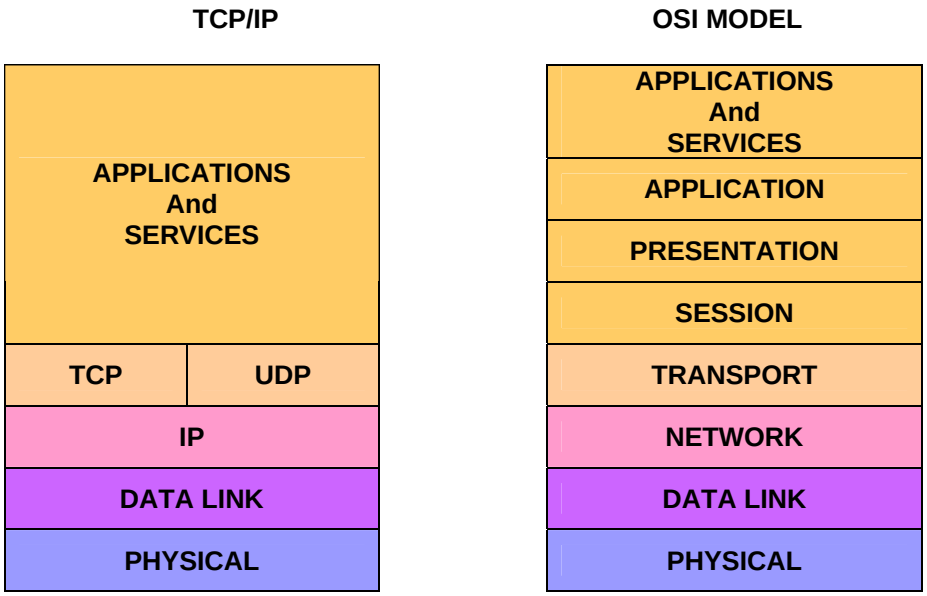

Figure 3.1 TCP/IP and OSI layers

The Physical layer

The Physical Layer (layer 1) of a network technology primarily defines the hardware it uses. Physical layer standards include detailed descriptions of media, such as coaxial cable, twisted pair or fiber. They specify physical attachments and signalling methods.
An SNMP management station can trigger tests that check the status of a medium or of the interfaces to the medium.

The data link layer

The Data Link Layer (layer 2) is concerned with moving information between two systems connected by a point-to-point link, a LAN, or a packet network (e.g., frame relay) circuit. The Data Link Layer:

- Packages information into frames
- Provides physical address information identifying sources and destinations.
- Identifies the protocol type for the information – e.g., IP, Novell IPX, or Banyan Vines

An SNMP management station can be used to configure, activate, and deactivate interfaces. A management station can obtain incoming and outgoing frame, octet, and error counts for each interface.

The Network Layer

The Internet Protocol, IP, operates at layer 3, the Network Layer, it has the job of routing data across a network. Any network that is based on IP is called an *internet*. The correct operation of IP depends on:

- Assignment of IP network addresses
- The ability to translate IP networks addresses to physical ones
- Accurate entries in routing tables

An SNMP management station can check up on IP address assignments, address translation tables, and routing tables. It can obtain counts of incoming and outgoing IP traffic and errors.

The transport layer

Two protocols, TCP and UDP, operate at layer 4, the Transport Layer. The Transmission Control Protocol, TCP, sets up connections and is responsible for the reliable transmission of data from one application to another.

The User Datagram Protocol, UDP, delivers simple stand-alone messages from one application to another, in other words, UDP is a connectionless protocol. It is the preferred protocol for carrying network management messages.

An SNMP management station can watch the number and duration of TCP connections at a system, and track who is talking to whom. It can obtain counts and UDP traffic and errors.

Applications

Standard TCP/IP applications include electronic mail, file transfer, and terminal access.

Protocol data units

Any formatted message is called a Protocol Data Unit or PDU. The layer 2 formatted Protocol Data Units are called *frames*. The IP layer 3 Protocol Data Units are called *datagrams*. TCP's layer 4 PDUs are called *segments*, while UDP's layer 4 PDUs are called *user datagrams.*

3.2 SNMP MANAGEMENT INFORMATION

3.2.1 The model

SNMP views management information as variables and their values. SNMP operations retrieve or modify the value of variables.
A *Management Information Base (MIB)* is a collection of information that is organized hierarchically. MIBs are accessed using a network-management protocol such as SNMP. They are comprised of managed objects and are identified by object identifiers.

A class (or type) of management information is called an *object* or an *object type*. A specific instance from a class of management information is called SNMP *variable* or an *object instance*. The OBJECT-TYPE construct is used in MIB modules to define each class of management information and specifies an object's data type, the maximal allowed access, its assigned identity, how instances are identified, and its semantics(or behaviours). An object may be defined to have exactly one instance, or may be defined to have multiple instances. A *scalar object* has exactly one instance. A *columnar object* has zero, one, or more instances at any point in time.

To be consistent with columnar objects, all scalar objects use a mechanism to identify the one instance in each class. The mechanism is identical for all scalar objects. An example of a scalar object is the location of a managed system, since a system has exactly one location. An example of a columnar object is the state of a TCP connection. There may be zero, one, or more TCP connections to a system at any point in time. Each TCP connection that exists has a state.

The SMI requires columnar objects to be organized into *conceptual tables*. All the columnar objects in a conceptual table must use the same mechanism for identifying instances within their class.

3.2.2 Object Identifiers

An identification scheme from ASN.1 is used throughout SNMP to uniquely identify items for all space and time. An identifier in this scheme is called an *object identifier* (or OID) and the identity of an item is determined by its OID value.

An OID values is an ordered sequence of non-negative integers, written from left to right, containing at least two elements. This scheme was created by the ISO and CCITT (now the ITU) organizations. OID values are organized like a hierarchical file system and the IETF's domain name system (DNS). Administration of assignments is delegated. The top levels assignments are controlled by the ISO and ITU organizations, which put restrictions on the first two numbers in the sequence. Lower level assignments are delegated and may also be sub-delegated.

A rooted tree (as shown in figure 3.2) is often used to illustrate the numbers in the sequences that correspond to OID values. Each vertex in the tree is labelled with a number, such that a path from the root to the vertex corresponds to the sequence of numbers that is the OID value for the vertex.

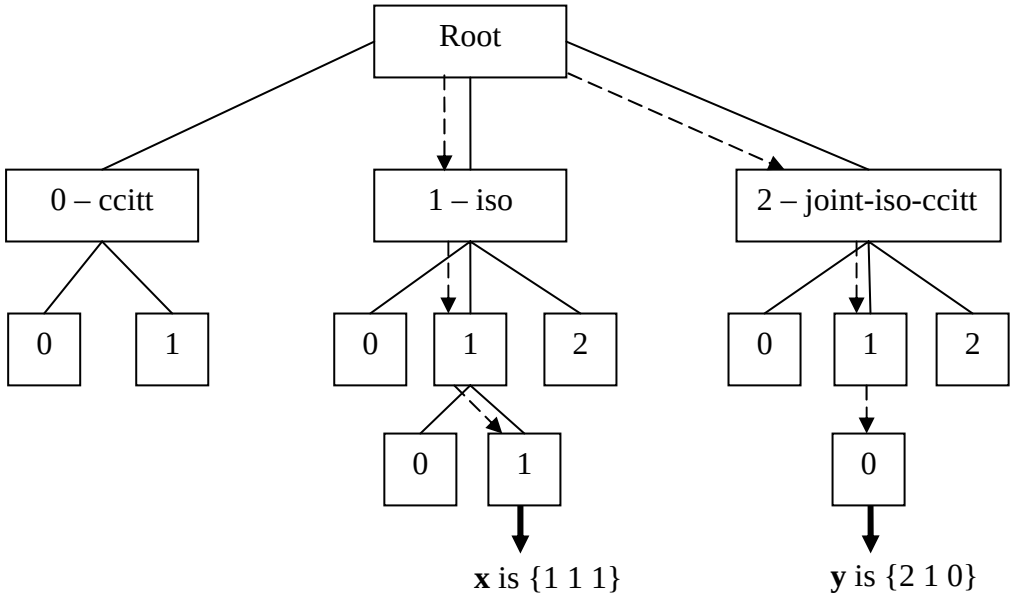

Figure 3.2 Object Identifier tree

4. SNMP MIB BROWSER REQUERIMENTS

4.1 Functional Requirements

The first objective of the project is to find a suitable storage structure for SNMP (Simple Network Management Protocol) MIBs (Management Information Base) in a relational database that will allow for fast retrieval of the management information.
A tool must be developed for MIB definition files to be parsed and for the information put into the new database structure. To parse the MIB files a specific Java library is needed, the proper library have to be chosen testing the efficiency of the existing ones.

To demonstrate the speed of information retrieval, a MIB Browser shall be developed. The MIB browser will allow a user to explore the MIB structure and query equipment for status.

The MIB browser shall be web based, developed in Java and deployed on a J2EE compliant web server (E.g. BEA Weblogic)

Graphical User Interface (GUI)

- The Graphical User Interface shall be web-based.
- The SNMP MIB MANAGER shall run on Internet Explorer Version 5.0 or later.
- The GUI design shall be plain and easy to use. Also shall be easy to understand the results.
- The application shall support all standard web techniques (scrolling, clicking, minimizing, maximizing, etc).
- The user shall only be allowed to select from a list of valid entries.
- A standard, consistent approach to notifying the user of errors and warnings shall be implemented.
- The application response of a query shall be as fast as possible (less than 3 seconds shall be acceptable).

Database Services

- SNMP MIB BROWSER/MANAGER shall be able to access data located in any database that supports JDBC.
- SNMP MIB BROWSER/MANAGER shall work with GNdB (the GNMS database).
- GNdB shall contain all the MIB information that SNMP MIB Browser needs in order to display all the queries.
- All the information that the GNdB returns will be in XML format.

Specifics

MIB MANAGER

- MIB Manager shall be able to add and delete SNMP MIB Modules into GNdB
- MIB Manager shall control if a MIB module is dependent on another one, and in this case, alert the user.

MIB BROWSER

- MIB BROWSER shall provide the capability to load and view MIB Modules in a MIB Tree.
- It shall facilitate traversing of the MIB tree
- It shall facilitate performing of the Basic SNMP Operations.
- It shall query the equipment by means of a SNMP WALK, showing the value of the object selected.
- MIB BROWSER shall changes the values of the Managed Objects sending a SET query

4.2 Non-Functional Requirements

This section defines appearance specific to SNMP MIB BROWSER.

Design

- There shall be a main page with 3 frames (as per the standard GNMS design structure)
- The top frame shall be the same as the GNMS standard structure (shall be a static frame).
- The left frame shall be the menu where the users can choose the different applications (shall be a static frame).
- The main frame shall be the frame where the applications will display. This frame shall be dynamic, and it will change every time the user chooses a different application.
- The main frame shall also display the results of the queries.
- The MIB Manager is accessed from the “Menu Frame” on the left hand side of the GNMS home page

Performance

- Storing MIB modules shall be optimised such that they minimize the processing load on the database.

- The information retrieval shall be optimised such that they minimize the response time for frequently used operations.

Audit

- All active actions executed in SNMP MIB Browser shall be recorded in a Log-file.

5. SNMP MIB BROWSER: A Java Application

In this chapter, the background elements to the project are discussed.

5.1 JAVA

The basic reason to develop the MIB Browser application in Java is that the GNMS application is implemented in Java and is part of the requirements.

The GNMS application is developed in Java because by definition, Java is simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, multithreaded, and dynamic.

Java is simple

In reality, no language is simple. However Java is considered a much simpler and easy to use object-oriented programming language when compared to other programming language like C++. Java doesn't use pointers and the inheritance are dealt by a simple structure called interface.

Java uses automatic memory allocation and garbage collection, his syntax makes Java programs easy to write and read. Also, being Object-oriented provides greater flexibility, modularity and reusability.

Java is distributed

Java incorporates routines to create network communications, Java programs can access to the objects across the network easily.

Java is interpreted

When the Java programs are compiled a Java bytecode is created, which is an executable for the Java Virtual Machine. The bytecode is machine independent and is able to run on any machine that has a Java interpreter.

Security and Reliability

Although the bytecode can be executed automatically in the machines, Java has a multilevel system of security:

- When a program is compiled it doesn't allow to access to the system resources, removing the pointers and memory allocation.

- The Java compiler ensures that the code is secure verifying that it doesn't do any of the following: forge pointers, violate access restrictions, incorrectly access classes, overflow or underflow operand stack , use incorrect parameters of bytecode instructions and use illegal data conversions.

- The Java interpreter ensures that the classes doesn't access the file system except if the client or the user allow it.

Java is Robust

Java compels the program to check for possible errors that would show up during execution time.

Java is Portable

Java has a platform independence, Java runs on most major hardware and software platforms.

Java is Multithreaded

Multithreaded is the capability of an application to execute more than one task at the same time.

Java is Architecture Neutral

Architecture neutral means that it is platform independent. A Java program can be run on any platform with a Java Virtual Machine (JVM). The JVM interprets the bytecode for a hardware platform so that it can perform the Java programs.

Java is Dynamic

During the execution of an application, Java can dynamically load classes that it requires either from the local hard drive, from another computer on the local area network or from a computer somewhere on the Internet.

5.2 MIB BROWSER IN JAVA

In this part I want to give a brief theoretical explanation of all the technologies that the MIB BROWSER uses, showing in some cases, the process that I followed to select one and not the other.

5.2.1 Servlet

At the beginning of the project I had to choose some methods implementation that my application will use and one of these choices was between a servlet application and an applet application.

The following table shows the main advantages and disadvantages of both structures:

Implementation Type	Advantages	Disadvantages
Servlet	• The most dependable solution. • Takes advantage of the full power of the core Java APIs • Portable across operating systems and across web servers • Offers better performance for workstations with slower CPUs. • Usually less load time on the browser than applets.	• More difficult to install and configure. • More load on the server can cause poorer performance than applets would. This could occur if the server's CPU is very busy or if the bytecode generated is greater than the size of the applet JAR file.
Applet	• Easy to install and modify. • Processing occurs at the workstation, releasing the server from the responsibility of processing the code. • Does not require any custom modifications to the server. • The small size of the JAR files provides quick applet load times. • Quick code generation can be achieved over slow connections by preloading applets.	• Some installations of NS6 and IE5.5 do not print applets. • Stability depends on stability of the client's web server • Although the JAR files are small, the browser must read the JAR file, which can cause poor performance over slow dial-up lines. This problem is solved by pre-loading applets. • Not compatible with browsers that do not have a Java VM such as NS v1 or old versions of Mosaic.

Table 5.1 Applet vs Servlet

The decision was easy to made, because:

- Servlets are **server-side** Java programs that are loaded and run within the framework of a web **server**. However applets are **client-side** Java programs that are loaded and run within the framework of a web browser. GNMS tends to liberate the client-side from as much responsibility as possible.

- The servlet can serve as the middleware between the client and database.

- Applets are much more difficult to use because of their security restrictions, one of the most important is that an applet can only make

socket connections back to its original machine. One alternative is a signed applet, but the mechanisms for obtaining signatures and ensuring compatibility between browsers is difficult at best.

The MIB Browser application collects info and stores it in the database, as we can see above the best solution to implement connections to the database is the servlet, furthermore the interface look is not an important issue in our application then the applet could be completely discarded.

After choosing the application as a server-side application, another issue had to be resolved, for every program that I was about to write I had to decide between using proper servlets or Java Server Pages (JSPs).

5.2.2 Java Server Pages (JSP)

Java Server Pages (JSP) is a technology based on Java that has the feature of allow embedding HTML, in that way is easier to create the Web Pages with dynamic content.

The advantages of JSP are:

- **vs. Pure Servlets.** The Servlets can do the same things than the JSP can do, it is possible too to add HTML but printing lots of statements , then they are more inflexible and difficult to develop.

- **vs. HTML using JavaScript.** In Javascript is possible to generate dynamic HTML but only on the client-side, it can't access to the databases that are located on the server, in our application we need to generate dynamic HTML getting data from the database.

As we can see there is a different philosophy between JSP and servlets. At the end of the day they are different ways of using "Java and HTML". In servlets the emphasis is more on Java while in JSPs the emphasis is more on HTML.

In the MIB BROWSER application JSPs are used as all is based in HTML responses, then all the presentation screens that are not in static HTML should be in dynamic JSP.

5.2.3 Hyper Text Mark-Up Language (HTML)

SNMP MIB Browser as any web-based application uses HTML language. Even though we talk about JSP and servlets, an important part of the project relies on the HTML language. In fact, we use HTML code in several places:

- In JSP applications, where static HTML is mixed with dynamical code.
- We also use HTML with XSLT files to represent information directly from the database.

So, we can say that the HTML code is present in all the main parts of the SNMP MIB BROWSER application.

5.2.4 Oracle

GNMS, in general, uses an Oracle database to manage all the B&SC network information. This database is called GNdB and provides data for the GNMS subsystems. Some subsystems may also write data to the GNdB.

All the queries are made in Structured Query Language (SQL), which is the set of command that all programs and users must use to access data within the Oracle database.
SQL is a data sublanguage that stores updates and retrieves data in a relational database by means of statements, these statements are instructions to the database, the basic instructions are SELECT, INSERT, UPDATE, DELETE, CREATE, and DROP, using them we can create complex statements that retrieve a specific data from the database.

SQL makes easy to manipulate the data stored in the relational database, it allows sending a block of statements to Oracle at any time.

The way that GNMS manages all its queries to the database is by using the Enterprise Java Beans (EJB) architecture, which enables transaction processing and controls transactions across multi-platform enterprise systems and database servers.

5.2.5 Enterprise Java Beans (EJB)

Enterprise Java Beans (EJB) technology is the server-side component architecture for Java Platform, EJBs are reusable Java components that enable to develop component-based distributed applications, they reside in a EJB container that provides a standard set of services like security, persistence, concurrency , transaction management, locking and environment.

The figure shows the main parts of the EJB structure:

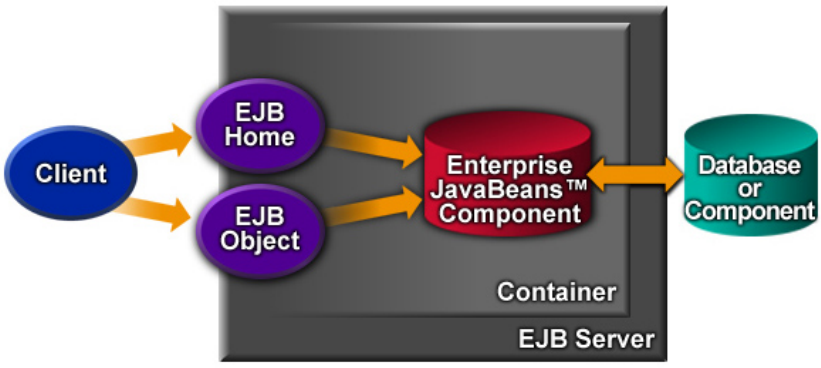

Figure 5.1 EJB structure

The client cannot instantiate an EJB Object directly, the EJB container intercepts all methods calls made by the client and it is the responsible for authorizing the access and carries out the transaction.

Each EJB provides some interfaces that its clients use to work with the EJB:

- Remote Interface: is the business end of the EJB. This is the set of actual services provides by the EJB.
- Home Interface: provides life cycle methods for creating, destroying, and locating beans.

The following steps show how to build EJBs:

- Step 1: Install an Enterprise Java Beans Server. In GNMS we use Weblogic Server 6.1.
- Step 2: Specify the Enterprise Java Beans Remote Interface.
- Step 3: Specify the Home Interface.
- Step 4: Write the Enterprise Java Bean Class.

When any GNMS client application tries to make a query in the database, the EJB server uses a container that will manage all the process.
This container calls the Bean program that connects to the database and automatically makes the query.

The database returns a ResultSet (from Java JDBC) to the Bean. The Bean transforms it into an XML document and passes this document to the container. From the container goes back to the initial GNMS client application.

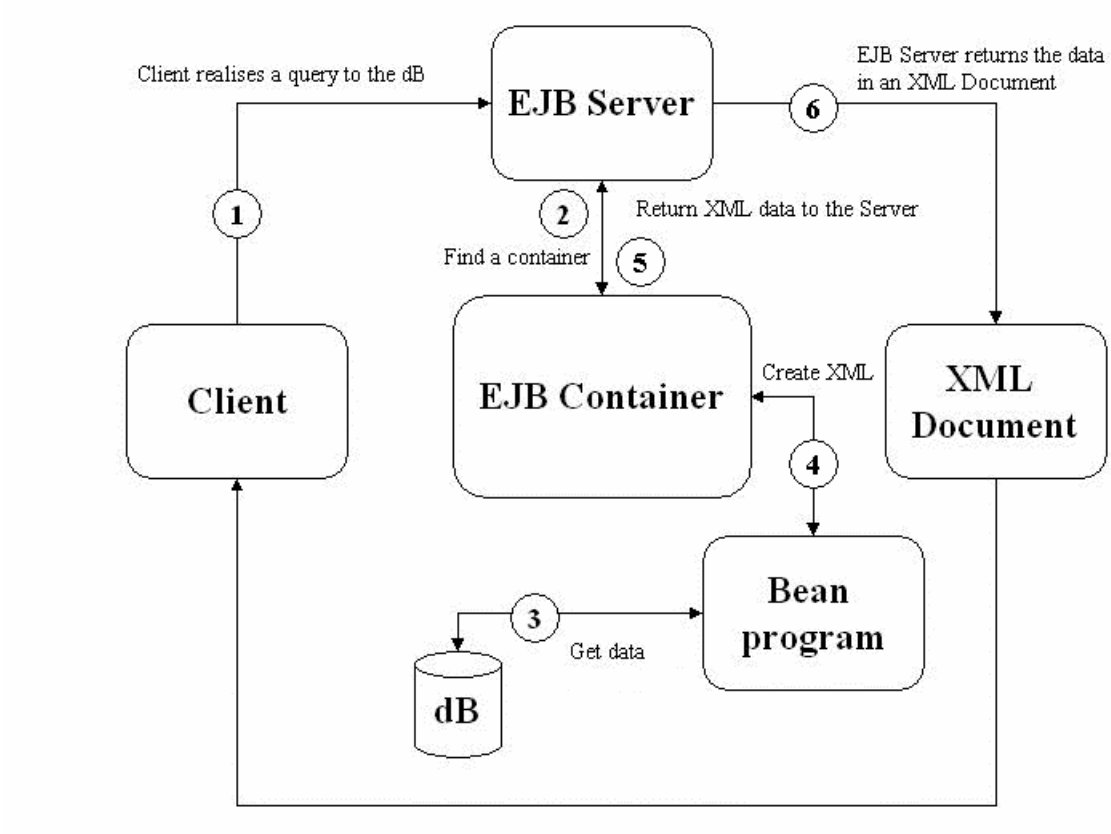

Figure 5.2 . EJB Diagram

The EJB container is the most import part of all this process. GNMS has the follow container configuration:

- As Home Interface: DBServiceHome program.
- As Remote Interface: DBService program.
- As an EJB Bean class: DBServiceBean class that contains all the possible GNMS database queries.

5.2.6 eXtensible MarkUp Language (XML)

Extensible Markup Language (XML) is a language subset of SGML (Standard Generalized Markup Language). XML defines a simple way for structuring data, allowing interchange of information between applications.
Java and XML are a good combination for developing web applications used for different clients that use information from various servers.

One of the most important feature is that it can be read and write using a simple text editor. As such, XML satisfies two compelling requirements:

- Separating data from presentation.
- Transmitting data between applications.

Any GNMS application that makes a query into the database will receive an XML document with all the data. This process consists of a set of Java classes that:

- Process any SQL query (using EJB).
- Pass that query to the database (using EJB as well)
- Return the results of the query as a ResultSet of Java JDBC to the EJB.
- Transform the ResultSet in an XML document using the EJB application.

5.2.7 eXtensible Style-sheet Language : Transformations (XSLT)eXtensible

XSLT which stands for eXtensible Style-sheet Language: Transformations, is a technology which is primarily designed for transforming one XML document in another one.

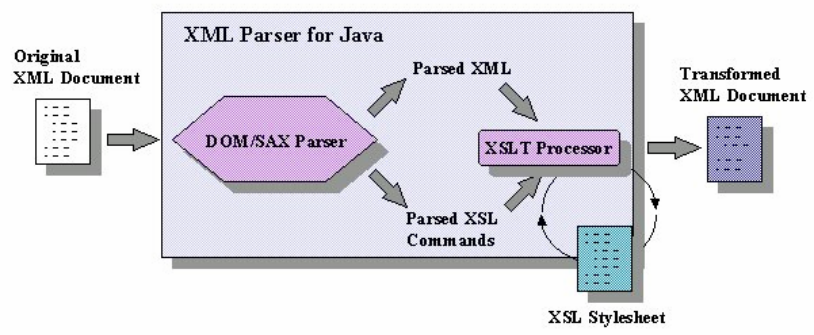

Figure 5.3 XSLT functionality

However, XSLT is capable of transforming XML to HTML and many another text-based formats, so a more general definition might be: XSLT is a technology for transforming the structure of an XML document.

Converting XML to HTML for display is probably the most common application of XSLT today, and it is the one I use in all the programs in my project. Once you have the data in HTML format, it can be displayed on any browser.

In SNMP MIB BROWSER application I use the XSLT style structure to represent some data directly from the database. The way that this works is really easy to understand:

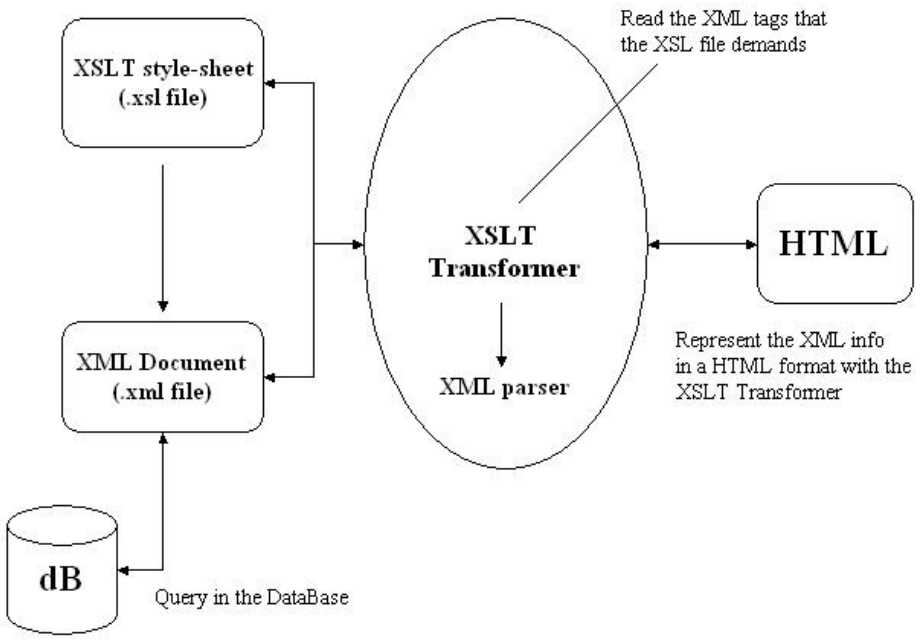

Figure 5.4 . XSLT Transformer

- When we create the XML in the dBServiceBean program, we add a header with a link to the XSLT style-sheet.
- Now, we just need to create a XSLT program that gets the tag names and the tag info and display them as HTML tables. In other words, the XSLT transform the XML information in HTML information to display in the browser.

The transformation process has two main aspects:

- The first stage is a structural transformation, in which the data is converted from the structure of the incoming XML document to the structure that reflects the desired output.
- The second stage is formatting, in which the new structure is output in the required format (as HTML in our case).

5.3 SOFTWARE PLATFORM

All the computers in the OSS intranet have Windows NT as a main Operative System. Due to this all the software used in my project run under a Microsoft environment.

The most important programs that I used were:

5.3.1 Jdeveloper 3.2.3

Figure 5.5 JDeveloper Start Screen

5.3.1.1 Introduction to JDeveloper

JDeveloper is a full-featured, integrated development environment for creating multi-tier Java applications. It enables you to develop, debug, and deploy Java client applications, dynamic HTML applications, web and application server components and database stored procedures based on industry-standard models.

The following diagram shows the logical architecture of applications built with JDeveloper:

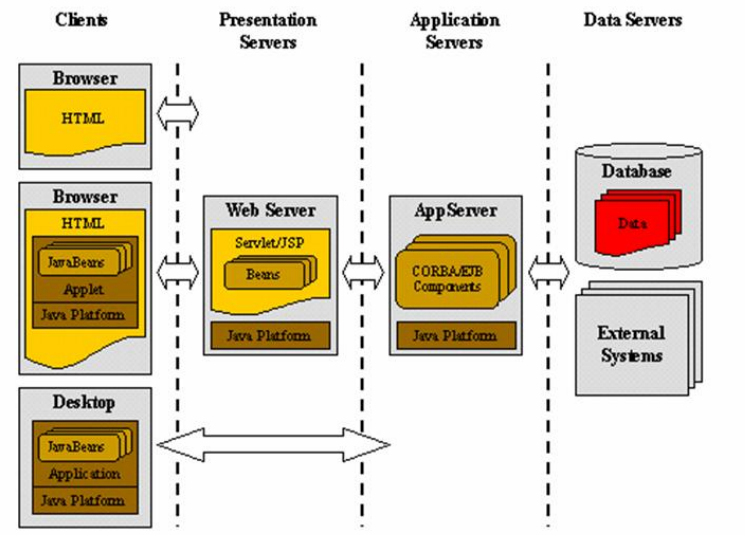

Figure 5.6 Logical Architecture of Applications Built with JDeveloper

To help the developer to implement this application architecture, JDeveloper provides powerful features in the following areas[6]:

Oracle Business Components for Java

Oracle Business Components for Java is an application component framework that provides reusable building blocks of software that manage all the common facilities required to:

- Author and test business logic in components which integrate with relational databases
- Reuse business logic through multiple SQL-based views of data
- Access and update the views from servlets, JavaServer Pages (JSPs), and thin-Java Swing clients
- Customize application functionality in layers without requiring modification of the delivered application

The benefits of using the Business Components framework are reduced development cost and time.

It is important to note that behaviour provided by the Business Components framework can be easily overridden in your domain-specific components with a few lines of code.

Web Application Development

With JDeveloper, you can build and deploy server-side Java applications that deliver dynamically generated HTML to any client running a browser.

[6] Oracle JDeveloper Feature Overview

Java Servlets

Servlets are portable Java classes that service HTTP requests and dynamically generate HTML. Servlets are portable across platforms and tiers and can be scaled. Servlets include integrated support for running in a multithreaded environment. JDeveloper provides wizards to create servlets and an integrated servlet engine to test and debug your servlets.

JavaServer Pages

JavaServer Pages (JSP) technology separates the user interface from dynamic content generation, JSP technology supports a reusable component-based design, making it easier and faster to build web-based applications.

JDeveloper provides a full array of tools for developing, debugging, and deploying JavaServer Pages. JDeveloper includes wizards to create JavaServer Pages. The JDeveloper JSP Element Wizard provides an easy way to add JDeveloper's data tags or predefined web beans to the code. JDeveloper also provides a wizard to create web beans.
These web beans can be reused in JSPs and provide a convenient way to package common HTML generation logic.

JDeveloper includes a complete JSP runtime engine. This allows JDeveloper to quickly launch the JSP for viewing or debugging within the integrated debugger.

Java Client Application Development

For developing Java-based clients, JDeveloper provides code-generation wizards and industry-standard data-aware controls.

Data-Aware Controls

JDeveloper includes a variety of data-aware controls that conform to the InfoBus industry standard. These controls provide for form-based access to the Oracle database without the need to write any JDBC (Java Database Connectivity) code. It is possible to add data-aware controls to a form by dragging and dropping from the JDeveloper tool palette.

Java in the Database

With JDeveloper, you can easily create Java stored procedures, Enterprise JavaBeans, and CORBA server objects. JDeveloper allows you to focus on the business logic, while it takes care of the CORBA and Enterprise JavaBean requirements.

Component-Based Development with JavaBeans

JDeveloper includes a full set of GUI-based tools which allow you to make intricate code element changes to any class, simply by filling out appropriate fields. The representation of code that the GUI-based tools manipulate is easy to understand and the generated code, although complex, is always error free. With the GUI-based tools, it is extremely easy to add, edit, or delete all of the important elements of any class, including fields, methods, properties, BeanInfo, and event sets.
JDeveloper provides wizards to help you get your JavaBean, BeanInfo, Property Editors, Customizers, and Custom Events started. Then you can use the Class Designer to make refinements. The Deployment Wizard makes deploying applications a simple task. The JDeveloper Component Palette allows you to add all your favourite JavaBeans to any page on the Palette. It is also important to note that when you create a JavaBean, JDeveloper generates only the class you request and does not generate hidden interfaces or base classes.

Simplified Database Access

The Oracle Business Components for Java framework provides lots of database access functionality. However, JDeveloper also provides tools for when low-level access to databases is needed.

Integrated JDBC

For interfacing with the Oracle database, JDeveloper provides several JDBC drivers and also allows developers the use of any third-party JDBC drivers.

Visual Integrated Development Environment

JDeveloper is a visual tool that includes the project navigator, code editor, debugger, and compiler. It also provides wizards, form designers, and property and event editors to automate many programming tasks.

The JDeveloper IDE provides the following features:

- **Two-way Technology** - The JDeveloper IDE employs two-way technology that keeps code and design synchronized as a developer works. No compilation is required, a change made in the designer is immediately reflected in the code. Similarly, changes to the code are immediately displayed in the Visual Designer.

- **Unicode-Enabled Compiler with Integrated Dependency-Checking and Obfuscation Support** - JDeveloper's advanced, unicode-enabled compiler automatically verifies dependencies in compilation, and error messages link to the section of code in question (in both standard Java and SQLJ).

- **Docking MDI Interface** - The JDeveloper development tool is a Multiple Document Interface (MDI), allowing developers to quickly switch among editors and designers. The Navigator, Property Inspector, and debugging message windows are dockable palette windows.

- **Visual Debugger -** The visual integrated debugger supports breakpoints, evaluations, watches, and a wide variety of control flow commands. The visual debuggers support both JDK 1.1 and Java 2.

- **Multiple Project Management and Libraries -** JDeveloper's Navigator allows developers to manage many projects, each with their own compilation, run, and debug properties, within one workspace.

- **CodeInsight** - When the editor window is open for coding, the JDeveloper IDE optionally displays coding tips, including parameter lists and class members. A package explorer shows all available packages on the current classpath.

- **JDeveloper Addin Application Programming Interface (API) -** The JDeveloper Addin API is a collection of Java APIs for extending the Oracle JDeveloper IDE. With the Addin API, developers gain access to the internals of JDeveloper, giving them the power to extend the IDE with additional user-created tools or to provide integration with third party tools.

- **CodeCoach -** JDeveloper includes a set of tools that analyze the Java code and make suggestions on how to improve its quality and performance. CodeCoach analyzes the code while it is running and generates a detailed list of suggestions. You have full flexibility to include or exclude any suggestions for future invocations.

Java Language Support

JDeveloper generates 100% Pure Java enabling complete portability and interoperability by adhering to all Java language specifications including JDBC, SQLJ, Servlets, JSP, EJB, JTS, RMI, JARs, and serialization.

JDeveloper is compatible with the Java 2 SDK version 1.2 and supports JFC/Swing. However, since multiple JDKs are available at present, it also provides a feature for switching between available JDKs so that developers can build solutions for any environment.

5.3.1.2 Jdeveloper Tools used in SNMP MIB Browser

The last section presented a general overview to JDeveloper but I did not use all the tools it offers.

In particular I didn't use:

- Oracle Business Components for Java
- Simplified Database Access
- Component-based Development with JavaBeans
- Java in the Database
- Java Client Application Development

However I use JDeveloper as a:

- Visual Integrated Development Environment with very good results.
- Web Application Development to create frameworks for JSPs.

5.3.1.3 Jdeveloper User Interface

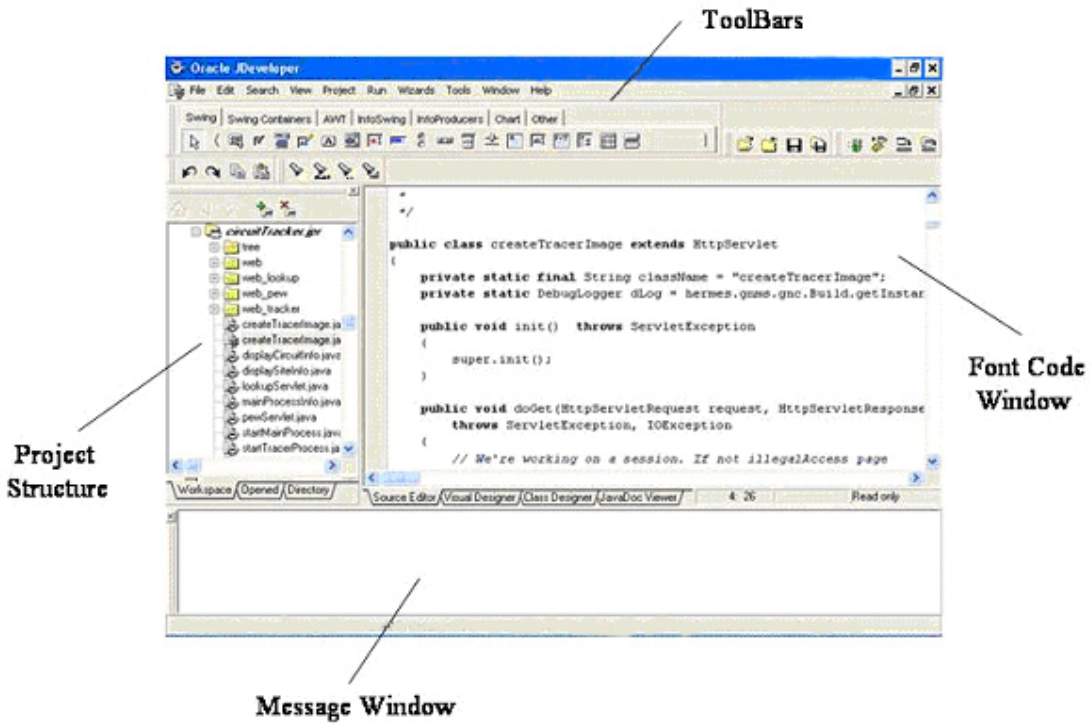

Figure 5.7 JDeveloper User Interface

The user interface, as shown in the figure above, incorporates a left-side pane that serves as a module and code browser, configured in a browse tree. The right-hand pane offers either a code or a display view of your project. The left-hand pane shows a file or class view of your project. It's possible to load more than one project into the project tree, which you may want to do if you keep the tiers in separate projects but want to debug and test them together.

JDeveloper has several wizards that speed the development of components such as a database access module or a JSP page. For example, to connect

to a database you go through several steps to specify the data source, select the database object, and choose which attributes you want visible from within your application. Even for a novice, configuring a database call in only a few minutes is possible.

Most development efforts will begin with an *n*-tier architecture design (UML modelling is slated for the next version of JDeveloper), which developers implement through the series of wizards. Once the basic structure of the architecture exists, the separate tiers can be parsed out to individual developers or development teams for coding. JDeveloper provides the ability to use the Oracle Repository for source code control for team development efforts. It's integrated into the IDE, so you can check your code in and out seamlessly. This makes it easy to start with the framework on a single machine, check it in, and make it available to the team.

5.3.2 WebLogic Server 6.1

Figure 5.8 . WebLogic Server Start Screen

BEA WebLogic Server[7] is a web and wireless application server, it implements the Java 2 Enterprise Edition (J2EE) platform specification, including Servlets, Java Server Pages (JSP), Enterprise JavaBeans (EJB), Java Messaging Services (JMS), and other platform services that provide the scalability, flexibility, and reliability required by multi-tier, business applications.

WebLogic Server contains Java 2 Platform, Enterprise Edition (J2EE) technologies and it consolidates Extensible Mark-Up Language (XML) technologies applicable to WebLogic Server and XML applications based on WebLogic Server.

WebLogic Server provides essential features for developing and deploying e-commerce applications across distributed, heterogeneous environments. These features are the following:

- Variety of clients options: supports clients that use HTTP, RMI (Remote Method Invocation) or IIOP (Inter-ORB Protocol), and mobile devices that use WAP (Wireless Access Protocol).

- Flexible Web services: it provides a platform to deploy Web Services as components of a application.

[7] BEA WebLogic Server

- Enterprise e-business scalability.

- Robust administration: it offers a Web-based Administration Console for configuring and monitoring the services including a command-line interface for configuration.

- E-commerce security : WebLogic Server provides Secure Sockets Layer (SSL) support for encrypting data transmitted across WebLogic Server, clients, and other servers. The user authentication and authorization for all WebLogic Server services is necessary as part of the security.

- Development and deployment flexibility

-

WebLogic Server Application Architecture

WebLogic Server is an application server, a platform for developing and deploying multi-tier distributed enterprise applications.
WebLogic Server works in the middle tier of a multi-tier architecture, this determines where the software components are executed in relation to each other and to the hardware, network, and users.

o ***Software Component Tiers***

The software components of a multi-tier architecture consist of three tiers:

- The ***client tier*** contains programs executed by users. The Web Browsers can request pages from WebLogic Server using the HTTP protocol and they can also access to the WebLogic Server services by deploying servlets and JSP pages in it. The clients access to the WebLogic Server services using standard J2EE APIs.
- The ***middle tier*** contains WebLogic Server and other servers that are addressed directly by clients, such as existing Web servers or proxy servers.
- The ***backend tier*** contains enterprise resources, such as database systems, mainframe and legacy applications, and packaged enterprise resource planning (ERP) applications.

Client applications can access to the WebLogic Server directly or through another Web server or proxy server. WebLogic Server generally connects with backend services on behalf of clients.

The figure shows the three tiers of the WebLogic Server architecture:

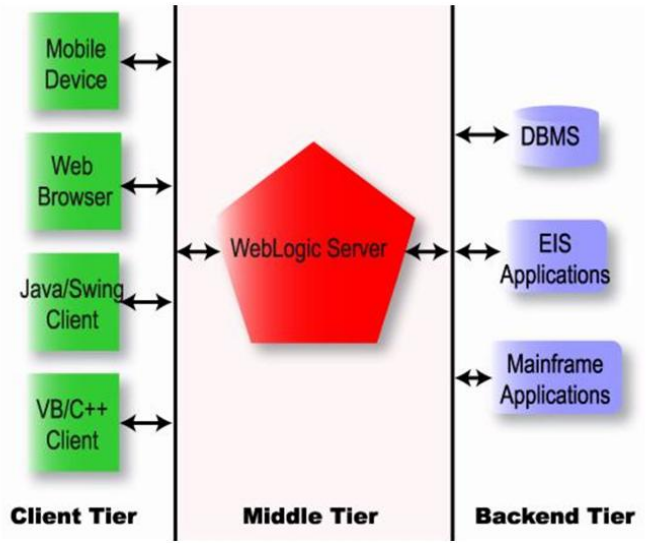

Figure 5.9 WebLogic Server Architecture

Application Logic Layers

WebLogic Server implements J2EE, the Java Enterprise standard. This functionality have the architecture of a layering of application logic, with each layer is deployed among WebLogic Server J2EE technologies. J2EE services include access to standard network protocols, database systems, and messaging systems.

The J2EEE components, including Servlets, JSP Pages, and Enterprise JavaBeans are executed in the WebLogic Server Web container

The web components provide the presentation logic for J2EE applications. The EJB components encapsulate processes and business objects. Web applications and EJBs are built on J2EE application services, such as JDBC, JMS (Java Messaging Service), and JTA (Java Transaction API).

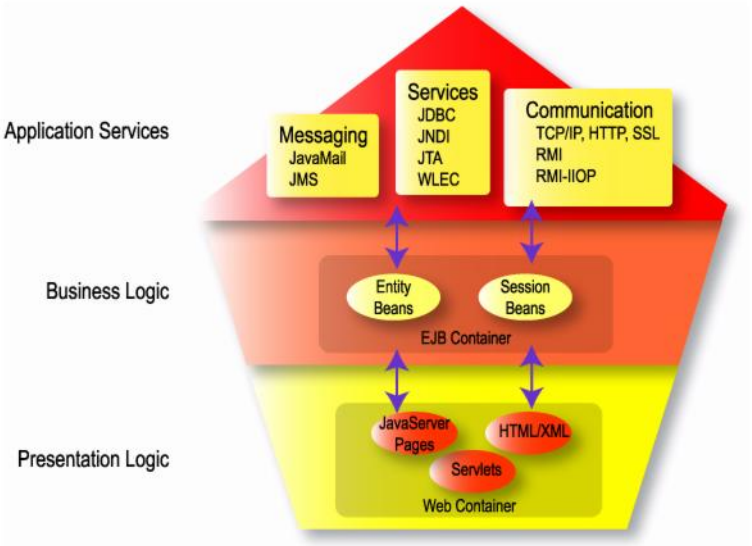

Figure 5.10 WebLogic Server Component Containers and Application Services

5.3.3 Microsoft Access 97

Microsoft Access 97 provides the capability to develop and manage relational database, GNMS use Oracle Database, but Microsoft Access 97 was used to create the new SNMP database for its simplicity in the way to develop it. Once the database was created, it was migrated to Oracle. When it was needed, TOAD was used to modify some objects (tables, indexes, constraints, …) , we can see an overview about TOAD in the next section .

5.3.4 Tool for Oracle Application Developers : TOAD 7.3

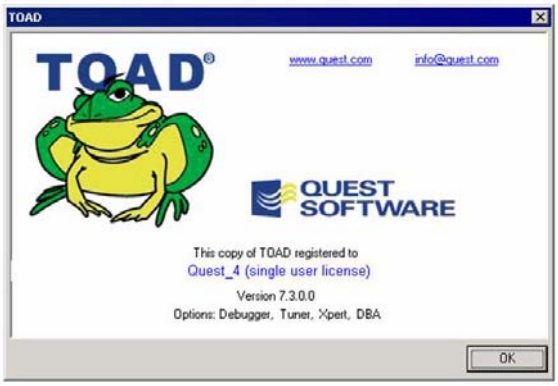

Figure 5.11 TOAD Start Screen

TOAD is a tool to manage databases that simplifies database administrations tasks, the GUI object browsers provide quick access to database objects. With TOAD the users can create and edit database tables, views, indexes, constraints, users, and roles easily. Thus it provides a SQL editor that enables to create and execute SQL statements.

The basic functionality of TOAD is[8]:

- Create, browse, or alter objects (tables, views, indexes, etc.) including Oracle8 TYPE objects
- Graphically build, execute, and tune queries
- Edit and Debug PL/SQL and profile "stored procedures" including functions, packages, and triggers
- Search for objects
- Find and fix database problems with constraints, triggers, extents, indexes, and grants

5.3.5 Others

I used other tools to develop my project such as:

- Internet Explorer 6.0 to run the GNMS application.
- Oracle SQL Plus 7.1
- XML / XSLT Editor.

5.4 LIBRARIES

5.4.1 Java 2 Platform, Enterprise Edition (J2EE) 2.3

Java 2 Platform, Enterprise Edition (J2EE) is a platform developed by Sun Microsystems that defines a standard for developing enterprise software applications.
The main features are:

- Portability across platforms and devices, allowing the developer to write the code only once and executing it on any platform.
- Scalability
- JDBC API for database access
- CORBA technology, incorporating the interoperability with other languages and enterprise resources.
- Security
- Support for Enterprise JavaBeans (EJB) components, Java Servlets API, JavaServer Pages and XML technology
- Quickly development and deployment

[8] TOAD for Oracle

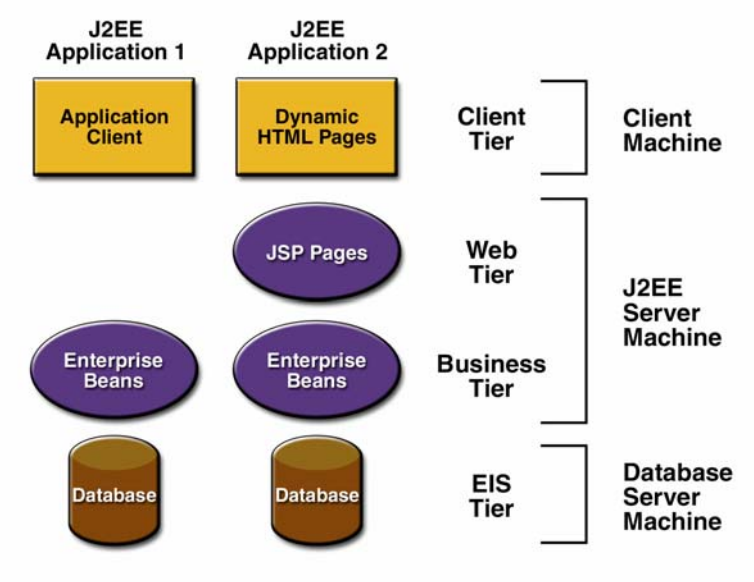

Figure 5.12 J2EE components

J2EE applications are made up of components (see figure above).

There are three types of J2EE component:

- Application clients and applets are components that run on the client.
- Java Servlet and JavaServer Pages (JSP) technology components are Web components that run on the server.
- Enterprise Java Beans (EJB) are server-side components based by containers that act as intermediary between client and server.

A J2EE component is a software unit that consist of sets of compiled Java classes that can be used for other components, we could confuse the standard Java Classes and the J2EE components but the difference is that the last one are assembled making up a J2EE application in compliance with the J2EE specification and they are run and managed by the J2EE server

GNMS uses this library to take advantage of its components. As we said before, all the GNMS database queries are handled by EJB. Using J2EE we can handle the EJB easily as we can see in the following figure:

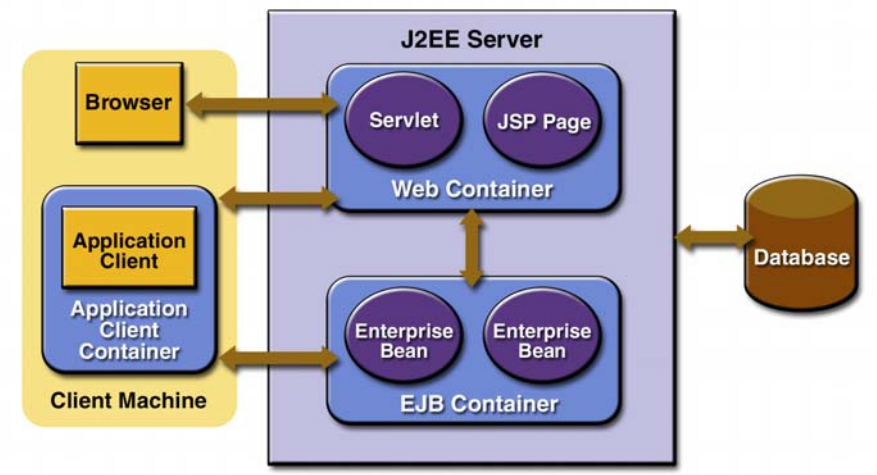

Figure 5.13 J2EE functionality

5.4.2 AdventNet SNMP

The SNMP MIB Browser is implemented using the AdventNet SNMP library. This library is used to get and manage all the information from the MIB modules.

But why need to use this library?

The MIB modules constitute a kind of database, the MIB modules are files in a text format, the only way to get the information and the relationships between objects is "reading" the module, a big number of functions and classes would be necessaries to walk the files to get the information, the development of these functions and classes is not a difficult task but it would take a lot of time, then the best solution cost-effective is to use a library.

The AdventNet SNMP API is Cross-Platform, scalable, flexible and it conforms to the RFC specifications.

The key features of AdventNet SNMP API are[9]:

- **Multi-lingual support:** Support for SNMPv1, SNMPv2c and SNMPv3.
- **Security**
- **MIB Parser:** Parse the MIB files.
- **MIB Loading:** Load MIB definitions.
- **IPv6 (Internet Protocol Version 6) support:** Provides connectivity with IPv6 and IPv4 based devices.
- **SNMP Broadcasting:** Broadcasts snmp packets across the network.
- **SNMP Beans:** Provides bean components
- **Database support:** Provides scalability by storing MIB definitions and SNMPv3 configuration data in any relational database such as, MySQL and Oracle.

[9] AdventNet SNMP API

- **MIB Browser:** Tool for administering network and system components.
- **Command line utilities:** Perform SNMP operations such as, SNMP GET, SNMP GETNEXT, SNMP SET, SNMP BULK, SNMP WALK, etc. on remote agents.

There are several modules available as part of the AdventNet SNMP API distribution, but only the *Low-Level API Architecture* and the *MIBs API* are used in the SNMP MIB Browser.

LOW-LEVEL API

The low-level API implements the basic functions of the SNMP protocol in Java classes grouped together into different packages. They provides the functionality to communicate with the devices, sending requests and receiving responses, allowing to configure the SNMP message level security and access control. In order to work with the different SNMP data types the API provides one classes for each data type that we can find.

MIBs API

The MIBs API is used to get and manage all the information from the MIB file. It consist of a collection of classes that allow to parse the MIB Module getting the MIB objects as a instances, in that way facilitates to access to the properties of the object (syntax, access-type, status,…) using specific functions.

Getting the MIB module information

To get the information from the modules it is necessary to follow the next steps:

1. Load and parse the MIB Module
2. Access to the nodes from a MIB tree
3. Get the leaf nodes information

Loading MIBs

There are two ways to load the MIBs, load the MIBs directly and from a compiled file. The main advantage of compiling first the file is that reduce the time of load, for that reason we decided to use this option. In that case when we load the module the application parses it and generates two new files :

- cmi - This file type contains MIB information, such as MibNode, MibModule, naming hierarchy, etc.
- cds - This file type contains the description and reference of the MIB nodes.

These files contains all the MIB Module information, if the same MIB module is loaded again the application doesn't parse it and the MIB is loaded from the cmi and cds files.

The *MIBOperations* is the class that provides the methods to load the MIB modules.

The following code show how to use *MibOperations* to load MIB files from a compiled file

```
MibOperations mibops = new MibOperations();
try {
   setLoadFromCompiledMibs(true)
    mibops.loadMibModule("RFC1213-MIB");
}
catch (Exception ex){
    System.err.println("Error loading MIBs: " +ex);
}
```

Accessing Node information

Once we load the MIBs files using the *MibOperations* class, the *MibModule* instances are created, one for each MIB file that we load. This instances contains all the MIB objects from the file, this objects are represented by *MibNode* instances that are MIB nodes in a MIB Module tree.

Some methods are present both in *MibModule* and in the *MibOperations*, for example *getMibNode()* , if we use *MibOperations.getMibNode()*, it will search for the node over all the MIB files that we are loaded, on the other hand if we use *MibOperations.getMibNode()* it will search only in the specific MIB Module file.

The *MibNode* class has methods to get all the details about a MIB variable and contains a instance that represents the syntax of the leaf node, the *LeafSyntax* class, this class can be used to get the data type and value of the *MibNode*. The *MibNode* and the *LeafSyntax* classes provide the methods necessary to access the information on the nodes in the MIB file.

In the next code is shown how to get the information of the MIB Module objects from a specific MIB file:

```
MibOperations mibops = new MibOperations();
try {
    mibops.loadMibModule("RFC1213-MIB");
    mibops.loadMibModule("XSC-MIB");
    mibops.loadMibModule("CISCO-SMI-MIB");

}
catch (Exception e)
{
    System.out.println("Exception : "+e);
    System.exit(1);
}
MibModule module =  mibops.getMibModule("XSC-MIB")
MibNode node = module.getMibNode("sysDescr");

System.out.println("Syntax:"+node.getSyntax());
System.out.println("Access:"+node.printAccess());
System.out.println("Status:"+node.getStatus());
System.out.println("Reference:"+node.getReference());
System.out.println("OID:"+node.getNumberedOIDString());
System.out.println("Node:"+node.getOIDString());
System.out.println("Description:"+node.getDescription());

LeafSyntax leafnode = node.getSyntax();
System.out.println("Value : "+leafnode.getType());
System.out.println("Data-type : "+leafnode.getEquivname());
```

In conclusion, to get all the information from a MIB module, first we need to create an instance of it (*MibModule*), and from it we will be able to get all the information necessary.

6. SNMP MIB BROWSER

This section explains why and how the SNMP MIB BROWSER was developed.

6.1 SYSTEM ARCHITECTURE

In this part is shown the old MIB BROWSER architecture and how it should be the new one.

Specifications

The first objective of the project was to find a suitable storage structure for SNMP (Simple Network Management Protocol) MIBs (Management Information Base) in a relational database that will allow for fast retrieval of the management information.

A tool must be developed for MIB definition files to be parsed and for the information put into the new database structure. A Java library will be used as a tool for the parsing.

To demonstrate the speed of information retrieval, a MIB Browser shall be developed. The MIB browser shall be web based, developed in Java and deployed on a J2EE compliant web server (E.g. BEA Weblogic)

6.1.1 Old MIB Browser

The following diagram shows the logical architecture of the old MIB browser system:

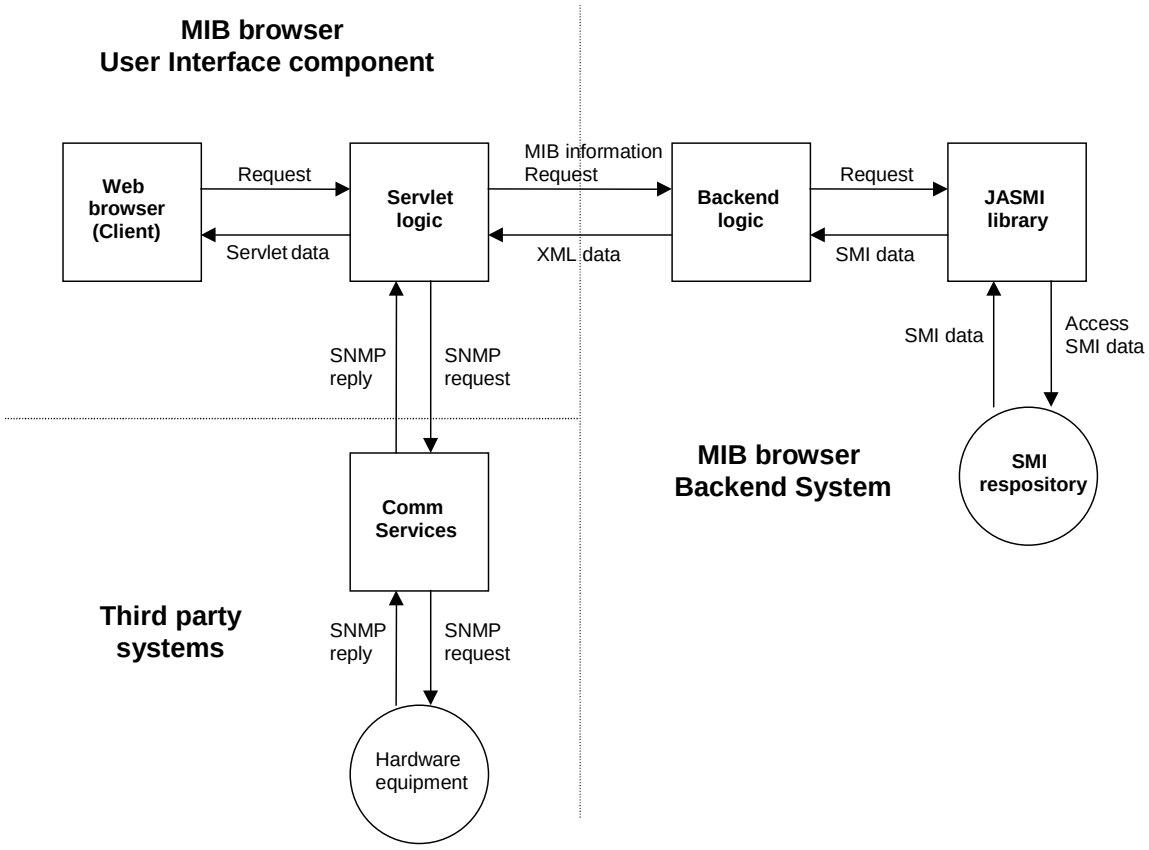

Figure 6.1 Old MIB Browser System

The system can logically be divided into the:

- MIB Browser User Interface Component
- MIB Browser Backend System

MIB browser User Interface Component

This component is responsible for:

(a) Displaying the MIB information for a piece of equipment. It will use the MIB browser system for this.

(b) Sending SNMP GET, GETNEXT, SET and SNMP “walk” requests to a piece of equipment.

MIB browser Backend System

This component is responsible for collecting and ordering MIB information from a SMI repository for the client. It uses a third-party library called JASMI (Java API for SMI) to do this.

That system is often a bit slow, all the variables information are kept in the SMI repository, every time we need to access to a variable it connects to the

repository and gets the information, it takes time to do this process, for that reason the new MIB Browser will storage all the MIB information into the relational database, achieving in this way more efficiency.

The JASMI library have a very complex and strict syntax, it is not easy to use. To store the MIB information to the database we would need to be able to access completely to the MIB module and get the information well sorted. For that reason, it could be necessary change the library, in the section 6.2, we test a few libraries, including the JASMI, to try to choose the one that more approach to our needs.

6.1.2 New Model

The new SNMP MIB Browser will replace the Backend System and the User Interface, it will not keep anything from the old one in these parts.

The Third Party System does not need any change, it works efficiently, for that reason this part will remain as it is in the old MIB Browser.

MIB browser Backend System

This component is replaced completely. One of the most important changes in the architecture of the Backend System is the creation of the database, where all the information from the MIB modules will be stored. In that way, the MIB modules have to be read and parsed only once, that involve a big efficiency with respect of the processing time.

First of all, the modules have to be load in the SMI repository, it means to store the MIB modules files in a directory on the server where they can be selected to store all the information in the new database.

All the files will be implemented with the new SNMP Java library.

MIB browser User Interface Component

This component is replaced totally for the new one too, the changes done are: use the new database to get the configuration objects and improve the look and feel of the interface.

To implement it the new library is used and it uses a JSP instead of the Servlet, we can see the reasons for this change in the section 5.2.2

With this component the user can manage all the SNMP MIB Browser application, such to load the modules in the repository, to store the information in the database, to get the information from the database, and to manage and control the equipments.

The new MIB browser architecture will be:

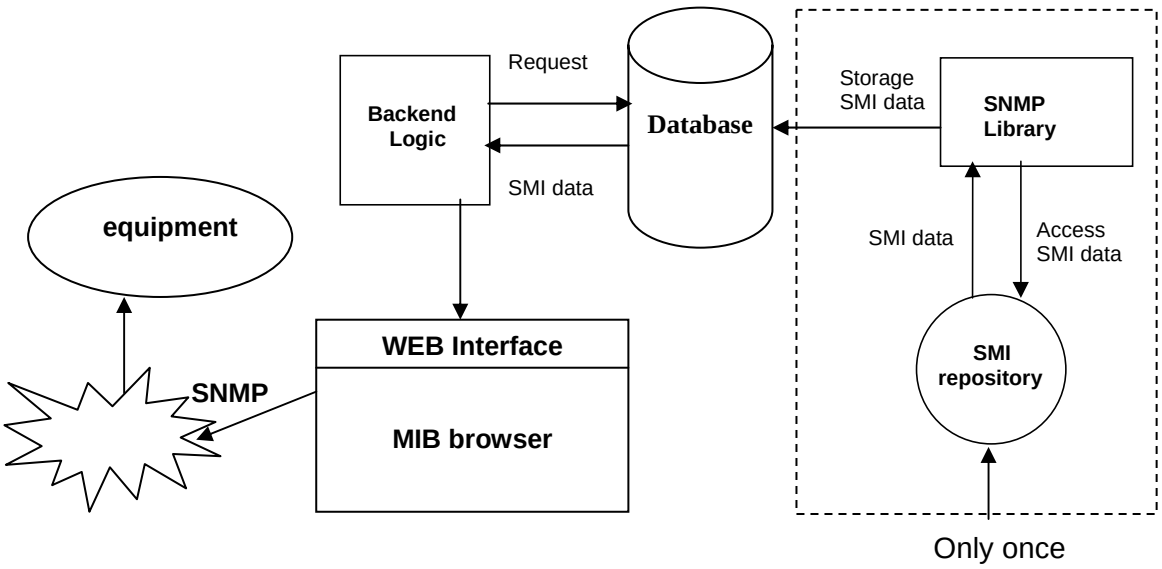

Figure 6.2 New MIB Browser Architecture

Basically we can split the project in three related parts:

- MIB Manager: The part that manage the MIB modules and the database.
- MIB Browser: The part where the user can manage and control the equipment.
- Database: It will contain all the information about the MIB modules.

6.2 SNMP LIBRARY

To read and manage the MIB modules a Java SNMP library should be used, this library will be used basically to store and to get the MIB data to the database, the first thing to do was testing different libraries to choose the optimal for our task.

6.2.1 Testing Libraries

We can find several JAVA libraries to manage the SNMP protocol: JASMI, AdventNet SNMP, jSNMP, Java SNMP (iReasonig Networks), WareMaker…The most interesting at first sight were, AdventNet SNMP, and jSNMP, but JASMI wasn't discarded directly because was used in the old MIB Browser System and it is necessary to verify if is worth to change it, all the old SNMP Java files from GNMS were implemented using JASMI.

The test consisted in load MIB modules to search the information (name, syntax, status, description and type) of one object by his OID in the modules loaded.

JASMI presented from the first a very strict syntax loading MIBs, we had to change some MIB modules syntax to load them (e.g. JASMI doesn't accept underscores in the modules), in addition the API's grammar is more difficult than the others two. JSNMP and AdventNet are very similar as easy to use and to understand.
Other aspects that were considered were: the run time and the used memory, as far as the time, we calculated how much time took loading the MIBs and find the object information.

The following table shows the results of the test made:

	AdventNet	**jSNMP**	**JASMI**
Average of the time loading modules (*ms*)	2156	753	2609
Minimum time loading modules (*ms*)	2141	718	1408
Maximum time loading modules (*ms*)	2172	828	2969
Average of time to find the information of the OID (*ms*)	<<0	<<0	87
Average of time to find the information of 10000 OID (*ms*)	23	18	-
Minimum time looking for information 10000 OID (*ms*)	16	15	-
Maximum time looking for information 10000 OID (*ms*)	31	31	-
Average used memory loading modules (*Kb*)	2228	3706	2821
Minimum used memory loading modules (*Kb*)	544	908	1408
Maximum used memory loading modules (*Kb*)	4876	6012	5000

Note: 50 samples were used to calculate all the averages

Table 6.1 Result of libraries test

Once the test was done, we discarded JASMI directly, to find the object information takes a lot of time and our application is based on lots of queries of this type. Furthermore, the code written took several lines up, meanwhile the codes from the other two libraries only took a few lines up.

In order to finish choosing which one was the better to use, we studied the other two libraries, the following table shows the key features of both libraries:

	Key features
AdventNet	• **Multi-lingual support:** Complete support for SNMPv1, SNMPv2c and SNMPv3. • **SNMPv3 security:** Support for HMAC-SHA-96, HMAC-MD5-96, CBC-DES and 128 bit AES encryption. • **Robust SMIv1 and SMIv2 MIB Parser:** Seamlessly parses the MIB definitions from any OEM vendor. Offers various flavors of parsing based on the MIB definitions. • **MIB Loading:** Option to load MIB definitions from a pre-compiled file, a Serialized file, or a Database to boost the performance. • **IPv6 (Internet Protocol Version 6) support:** Provides connectivity with IPv6 and IPv4 based devices. • **SNMP Broadcasting:** Broadcasts SNMP packets across the network to auto-discover SNMP devices in the network. • **SNMP Beans:** Provides high-level bean components such as, SnmpTarget, SnmpTable, SnmpPoller, TrapReceiver for easy application development. • **Database support:** Provides scalability by storing MIB definitions and SNMPv3 configuration data in any relational database such as, MySQL and Oracle. • **MIB Browser:** Tool for administering network and system components. Can be run as a stand-alone application or invoked from a web browser. • **Command line utilities:** Perform SNMP operations such as, SNMP GET, SNMP GETNEXT, SNMP SET, SNMP BULK, SNMP WALK, etc. on remote agents. • **License cost:** $995
jSNMP	• Reduces network traffic by combining requests from multiple clients into a single request and eliminating duplicate requests • User-specified caching threshold for requests • Eliminates data copying in ASN.1 encoding and decoding • Cross-platform • Accessible through Java, RMI, or CORBA • Small footprint • Speed on par with traditional C/C++ implementations • Complete SNMP v1/v2c/v3 support including trap/inform handling and authentication and privacy password alteration • Allows the user to communicate with network devices by specifying the Object Identifier (OID) such as the Basic Encoding Rules (BER) or Protocol Data Unit (PDU) format. • **License cost:** $995

Table 6.2 AdventNet vs. jSNMP

Both libraries are very similar, the most important differences between them are:

AdventNet

- it uses less memory loading modules
- his API is more clear and concise
- it provides more applications and functions than jSNMP

jSNMP

- it uses much less time loading modules

The modules have to be loaded only once, therefore this constant it is not very important.

We chose AdventNet for the applications and functions that provides, and for his clear and useful API.

6.3 SNMP-MIB OPTIMAL STORAGE

Once the Java SNMP library was chosen, we prepared for creating a database where to store all the MIB modules information.

The AdventNet SNMP library offers the option to store the modules in a database, but the structure of this database is not very efficient, of each MIB module loaded 14 tables are created, the amount of modules that we have to load is quite great, therefore the database would contain 300 tables.

We decided to create a new database structure following the relational model achieving in that way a database with a few tables.

6.3.1 MIB Modules Structure

As we have explained in previous sections, a MIB contains definitions of management information needed to manage a network. From a developer's perspective, it is difficult creating a MIB database from a high-level set of abstractions using traditional database technology. As shown, the MIB structure generates an information model consisting of lots of interrelated objects (Figure 6.3).

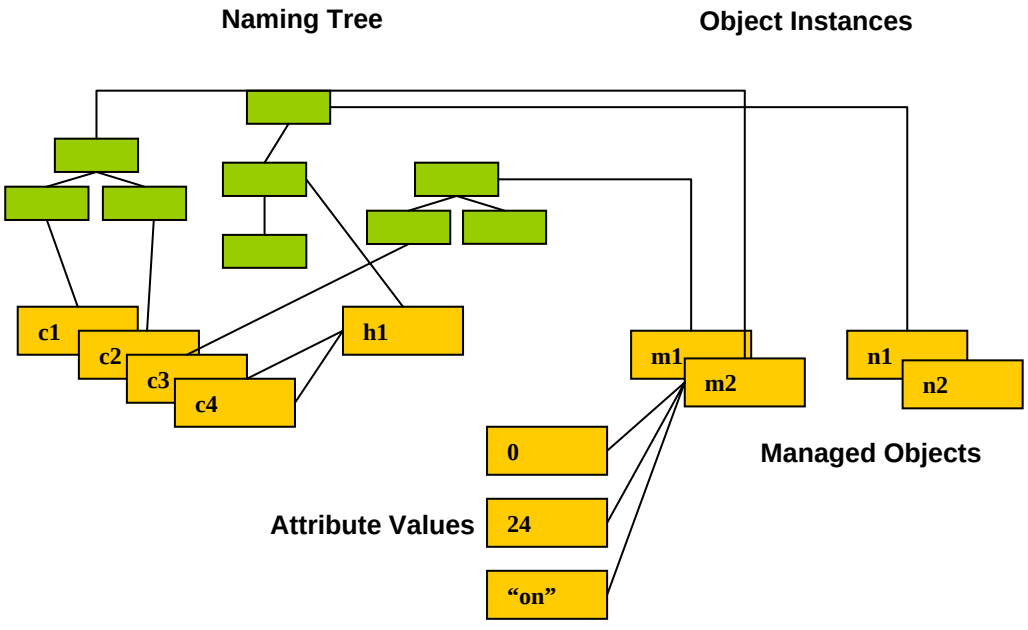

Figure 6.3: MIB interrelated objects

The next figure shows the difficulty mapping the MIBs structures into the relational database. A MIB tends to generate simple data descriptions because the objects are described in terms of columns and tables. Although this seems to be a simple task for relational databases, it does not correspond with the reality. Although the structure is all about tables, the real model of the equipment being monitored is a complex web of relationships.

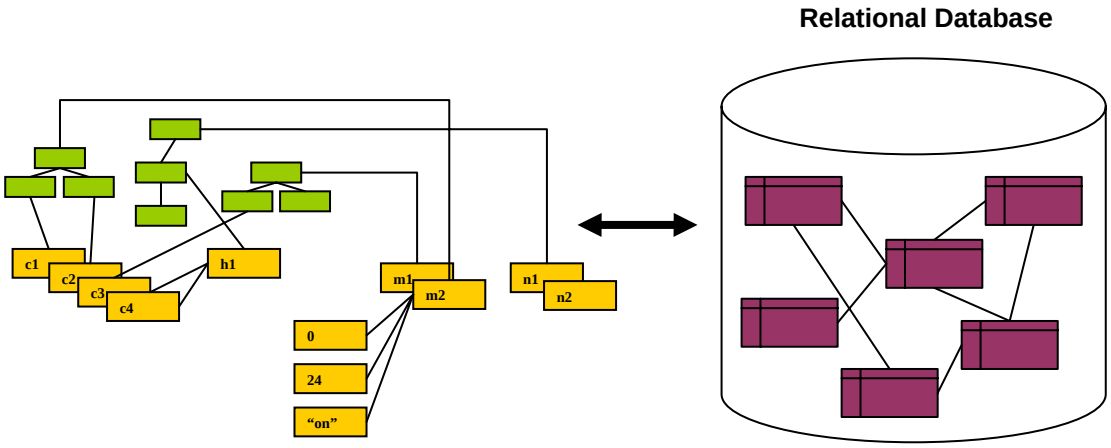

Figure 6.4: Mapping MIB structure into the database

6.3.2 Database Development

As we show in the previous section creating the database it was not an easy task. First of all the MIB files were explored to understand all the kind of objects that we need to store in the database and the relationships that they have among them, once the MIB modules were understood we proceeded to design the database tables.

The next figure show the database structure created, showing the relationships between tables:

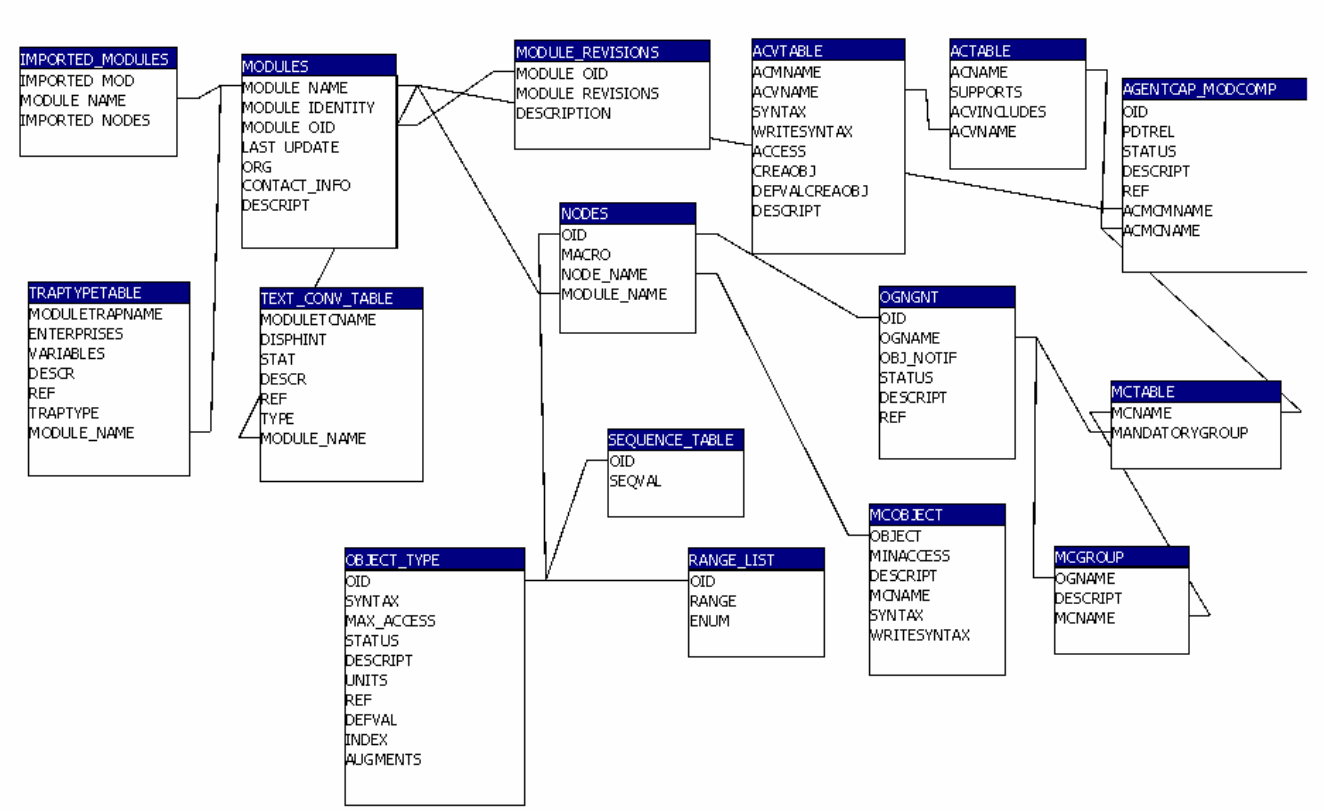

Figure 6.5: SNMP database structure

The following are the tables that the database created contains.

MODULES Table

It contains the information about all the modules loaded in the database.

Attributes

Field Name	Description
MODULE_NAME	The MIB module name
MODULE_IDENTITY	Specifies information about the MIB MODULE.
MODULE_OID	The Object Identifier
LAST_UPDATE	The Last Updated field - Specifies the date and time at which the module was created or last modified.
ORG	The Organization field - Gives information about the organization responsible for the module.
CONTACT_INFO	The Contact Info field - Specifies the contact information of the person responsible for the module.
DESCRIPT	The Description field - Explains the purpose of the module.

MODULE_REVISIONS Table

It contains the information about the module revisions.

Attributes

Field Name	Description
MODULE_OID	The Object Identifier
MODULE_REVISIONS	The Revision field - Specifies the revision timestamp in the UTC format.
DESCRIPTION	The description field- Explains the changes done to the module

IMPORTED_MODULES Table

Attributes

Field Name	Description
IMPORTED_MOD	Name of the imported module name.
MODULE_NAME	The name of the module
IMPORTED_NODES	Information on the imported objects.

NODES Table

Attributes

Field Name	Description
OID	The Object Identifier
MACRO	The name of the Macro type.
NODE_NAME	The name of the leaf node.
MODULE_NAME	The name of the module

OBJECT_TYPE Table

Attributes

Field Name	Description
OID	The Object Identifier
SYNTAX	The Syntax field - Specifies the syntax of the data type. It should be one of the basic SNMP data types.
MAX_ACCESS	The Access field - Specifies the allowed access to the leaf object.
STATUS	The Status field - Specifies the status of the definition.
DESCRIPT	The Description field - Describes the node or item being defined.
UNITS	Specifies the value of the Units field of the node object.
REF	The Reference field - Specifies the source of the definition.
DEFVAL	The Default Value field - Specifies an acceptable value for a columnar object which may be used to create an instance of a row.
INDEX	Specifies the value of the Index field of the node object.
AUGMENTS	Specifies the value of the Augments field of the node object.

SEQUENCE_TABLE Table

Attributes

Field Name	Description
OID	The object identifier of the table entry.
SEQVAL	The values in the Sequence construct.

RANGE_LIST

Attributes

Field Name	Description
OID	The Object Identifier
RANGE	The range value for the specified node.
ENUM	The enumerated values for the specified node.

OGNGNT Table

Attributes

Field Name	Description
OID	The Object Identifier
OGNAME	The name of this Object Group, Notification Group or Notification type
OBJ_NOTIF	The name of the objects that contains.
STATUS	The Status field - Specifies the status of the definition.
DESCRIPT	The Description field - Describes the node or item being defined.
REF	The Reference field - Specifies the source of the definition.

TEXT_CONV_TABLE Table

Attributes

Field Name	Description
MODULETCNAME	The Module Textual Convention field - Specifies the name of the textual convention construct.
DISPHINT	The Display Hint field - Hints on how to display the value of the type or describe the sub-structuring of the value.
STAT	The Status field - Specifies the status of the definition.
DESCR	The Description field - Describes of the node or item being defined.
REF	The Reference field - Specifies the source of the definition.
TYPE	Syntax of the base data type.
MODULE_NAME	The name of the module where the construct is defined

TRAPTYPETABLE Table

Attributes

Field Name	Description
MODULETRAPNAME	The Module Trap Name field - Specifies the name of the TRAP-TYPE construct.
ENTERPRISES	The vendor identification for the network management subsystem that generated the trap.
VARIABLES	Specifies one or more scalar or columnar objects with values describing the event.
DESCR	The Description field - specifies information describing the status of the variables.
REF	The Reference field - Specifies the source of the definition.
TRAPTYPE	Gives the name of the trap.
MODULE_NAME	The name of the module where the construct is defined

AGENTCAP_MODCOMP

Attributes

Field Name	Description
OID	The Object Identifier
PDTREL	The Product Release field - Describes the product release that includes the implemented capabilities.
STATUS	The Status field - Specifies the status of the definition.
DESCRIPT	The Description field - Describes the node or item being defined.
REF	The Reference field - Specifies the source of the definition.
ACMCMNAME	Agent Capabilities Module and the Module Compliance Module Name.
ACMCNAME	The Agent Capabilities or the Module Compliance Construct Name

MCTABLE Table

Attributes

Field Name	Description
MCNAME	The Module Compliance Construct Name
MANDATORYGROUP	The Mandatory Group field - Specifies a conditionally required object or notification group.

MCGROUP Table

Attributes

Field Name	Description
OGNAME	The Group Object
DESCRIPT	The Description field - Describes the node or item being defined.
MCNAME	The Module Compliance Construct Name

MCOBJECT Table

Attributes

Field Name	Description
OBJECT	The name of the Object
MINACCESS	Specifies the minimum allowed access to the leaf object
DESCRIPT	Specifies the reduction in the object's behaviours
MCNAME	The Module Compliance Construct Name
SYNTAX	The syntax field
WRITESYNTAX	The write syntax field

ACTABLE Table

Attributes

Field Name	Description
ACNAME	The Agent Capabilities Construct Name
SUPPORTS	Specifies the module identifier.
ACVINCLUDES	The AGENT CAPABILITIES VARIATION INCLUDES field - Specifies object and event groups that an agent implements.
ACVNAME	The Agent Capabilities Variation Name

ACVTABLE Table

Attributes

Field Name	Description
ACMNAME	The Agent Capabilities Module Name
ACVNAME	The Agent Capabilities Variation Name
SYNTAX	The SYNTAX field
WRITESYNTAX	The write syntax field
ACCESS	The Access field - Specifies the allowed access to the leaf object
CREAOBJ	The Creation-Requires field is used to document the objects that are required in a SET operation to create an instance of a row in a table.
DEFVALCREAOBJ	The DEFVAL clause is used to specify an acceptable value for a columnar object which may be used to create an instance of a row.
DESCRIPT	The Description field - Describes the node or item being defined.

The tool used to create the database was Microsoft Access 97, once the database was finished we proceeded to migrate it into Oracle. The migration was done with a specific tool used for the GNMS team.

The Oracle database was stored in a global database used for testing. The name of this database is *htest*.

It was time to implement the MIB Manager in order to store the MIB modules into the database, the MIB Manager is explained in the next section.

6.4 MIB MANAGER

The MIB Manager is the application that manages the MIB modules and the database. The user can load and delete the modules from the database and can store the modules into the repositories.

6.4.1 Location in the GNMS file structure

This application uses different programs to manage the database, always using the AdventNet library.

The main Java modules are in the */gnms/admin/mib*
The HTML and JSP modules are in */gnms/admin/mib/jsp* directory.

6.4.2 Development

The figure below represents the MIB Manager structure and contains all the classes used.

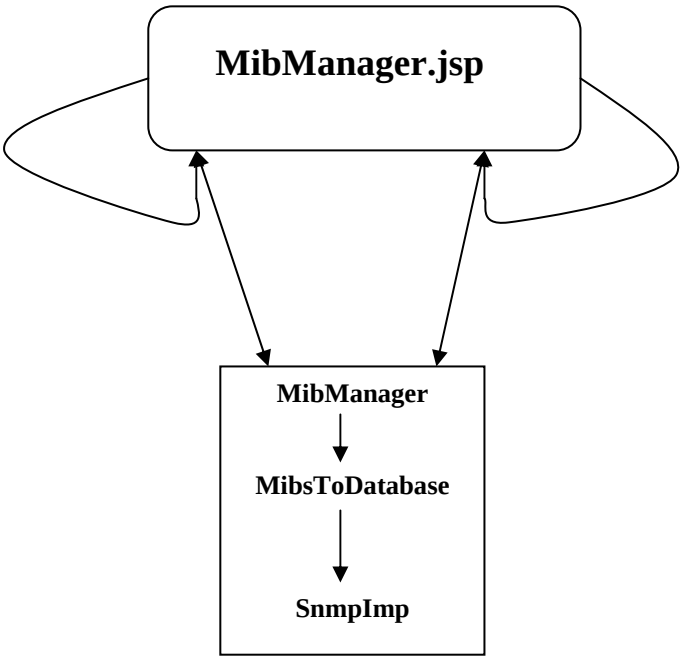

Figure 6.6 MIB Manager Classes

Here's a list of all of them:

MibManager: This is a JSP that displays all the modules loaded in the database. It contains two buttons for add and delete modules, if we add a module the same JSP is called and it displays the modules that we can add (the modules stored in the repositories). If we want to delete a module it displays the modules that we can delete (the modules already stored in the database). This JSP checks also if there is any problem adding or deleting modules and displays the error message.

MibsToDatabase: This is a class used to store and delete the modules in the database. It prepares all the information needed to do the query to the database.

MiscSnmp: This is a class that contains little functions that are very useful and we need to implement them very often. In order to not to implement every time the same code, for example to check one string, we create this class and when we need to use the function we only need to call it.

MibManager: This is the class that contains all the functions to add and delete the modules from the database. This class prepares all the information to do the query.

SnmpImp: This is a class used to query the database. It contains all the database queries, to add the modules, to delete them, to get all the information...etc.

6.4.3 Database Queries

This application uses several database queries to get the data needed, all are implemented in SnmpImp.java. These queries are:

checkModuleAdded*(impMod)*: checks if the modules imported in the module are stored in the database. If all of them are stored a boolean is returned with a value *true*, if not the boolean will have the value *false*.

setMibModules*(modName, modId, modOID, lastUp, org, contInfo, descript)*: inserts the information of the module in the MODULES table

setMibImportedModules*(impMod, modName, elements)*: inserts the information in the IMPORTED_MODULES table

setMibACMC*(Oid, prRel, status, descript, ref, acmcMod, name)*: inserts the information in the AGENTCAP_MODCOMP table.

setMibAgentCapab*(acN, supports, includes, varName)*: inserts the information in the ACTABLE Table.

setMibACVTable*(acMod, varName, synt, writSynt, acc, cObj, defVal,descr)*: inserts the information in the ACVTABLE table.

setMibModComp*(mcN, mandGroup)*: inserts the information in the MCTABLE table.
setMibMCGroup*(mcGroupName, descript, mcN)*: inserts the information in the MCGROUP Table.

setMibMCObject*(mcObjectName, minAccess, descript, mcN, syntax,writeSyntax)*: inserts the information in the MCOBJECT table.

setMibOrgGroupNotGroup*(oidOgNg, ogntName, objNotif, status, descript, refer)*: inserts the information in the OGNGNT table.

setMibNodes*(oid, macro, nodeName, modName)*: inserts the information in the NODES Table.

setMibObjectType*(oid, synt, maxAccess, status,descript, units, defval, augments, ref,indexTable)*: inserts the information in the OBJECT_TYPE table.

setMibRangeList*(oid, rang, enumeration)*: inserts the information in the RANGE_LIST table.

setMibSequenceTable*(oid, seqVal)*: inserts the information in the SEQUENCE_TABLE table.

setMibTrapTypeTable*(trapName, trapEnterprise, trapVar, trapDescript, trapRef, trapTrapType, modName)*: inserts the information in the TRAPTYPETABLE table.

setMibTextConvTable*(tCName, dispHint, status, descr, ref, type, modName)*: inserts the information in the TEXT_CONV_TABLE table.

deleteMibModule*(module)*: deletes the module from the database, deletes all the module information from all the tables.

6.4.4 Classes

Different classes are used to develop the MIB Manager application, the functions that are implemented will be shown below.

MibManager Class

This class is called for the MibManager.jsp which passes the name of the modules that have to be stored or deleted.

getMibModuleVector(): returns a vector that contains the name of the modules that are stored in the database.

addMod*(module)* : adds the module in the database calling the functions necessaries implemented in the class MibsToDatabase. If this module is on dependant of another one that is not store yet, it will return the name of the modules that have to be stored first.

deleteModule*(modules)* : delete the modules calling the function implemented in the class MibsToDatabase.

moduleFileEmpty*(module)*: checks before to add the module if the module have information or if is empty. There are some modules that doesn't contain any nodes, they contain only useful information.

MibsToDatabase Class

The main use for this class is to read the module information from the MIB files using the AdventNet library, it is used too to control if the imported modules are already stored and to delete the modules.

addMod*(module)*: load and parse the modules to be able to manage them, return the MibModule class which will be used to read all the module information modules.

modulesImportedNames*(String module)*: returns the name of the modules that are not added yet

deleteModule*(String module)*: call deleteMibModule from SnmpImp
The next functions read the MibModule class where all the module information is contained to get the information necessary to add it to the respective database tables:

setTableModules*(MibModule module)* : MODULES table, MODULE_REVISIONS table

setTableImportMod*(MibModule module)*: IMPORTED_MODULES table

setTableAgentCap*(MibModule module)* : AGENTCAP_MODCOMP table, ACTABLE table, ACVTABLE Table

setTableModComp*(MibModule module)*: AGENTCAP_MODCOMP table MCTABLE table, MCGROUP Table, MCOBJECT table

setTableOgngnt*(MibModule module)*: OGNGNT table

setTableObjectIdentifier*(MibModule module)*: NODES Table, OBJECT_TYPE table RANGE_LIST Table, SEQUENCE_TABLE table

setTableTrapTypeTable*(MibModule module, String mOID)*: TRAPTYPETABLE table

setTableTextConvTable*(MibModule module, String mOID)*: TEXT_CONV_TABLE Table

6.4.5 Code Structure

MibManager JSP Implementation Code

Pseudo code

```
// The main MIB Manager page

Read the parameter "added"

//The first time the MibManager is launched added = no
if( added.equals('no'))
      //Obtain the modules stored in the database
      call getMibModuleVector() from MibManager.java

      Show the name of all the modules in the browser
      "Add New" button→ on Click call MibManager.jsp with added= listMod
      "Delete" button→ on Click call MibManager.jsp with
                       added=delete

else if (added.equals ('delete'))

      //Obtain the modules stored in the database
      Call getMibModuleVector() from MibManager.java

      Create an html listbox with the name of the modules
      "Delete" button→ on Click call MibManager.jsp with
                       added='delete' and modules= the name of the
                       modules checked in the list box

else if (added.equals ('deleteMibModule'))

      Read the parameter "modules"
      Call deleteModule(modules) from MibManager.java
      Display on the screen that the module/s have been deleted
      successfully

else if (added.equals ('listMod'))

      Read the modules that are in the repository
      Display the names in an html listbox
      "Add" button→    on Click call MibManager.jsp with
                       added='addMibMod and modules= the name
                       of the modules checked in the list box

else if (added.equals ('addMibMod')

      Read the parameter "modules"
      Call addMod(modules) from MibManager.java

      if( all the imported modules from the module are stored)
            Display on the screen that the modules have been stored
            successfully

      if not
            Display on the screen a message that shows the name of
            the modules that we need to add before this one.
```

Graphical Algorithm

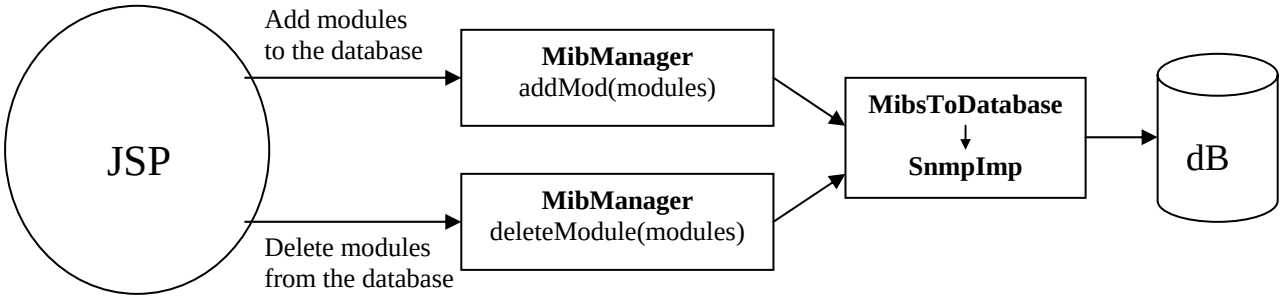

Figure 6.7 MibManager JSP Algorithm

Relationship between classes

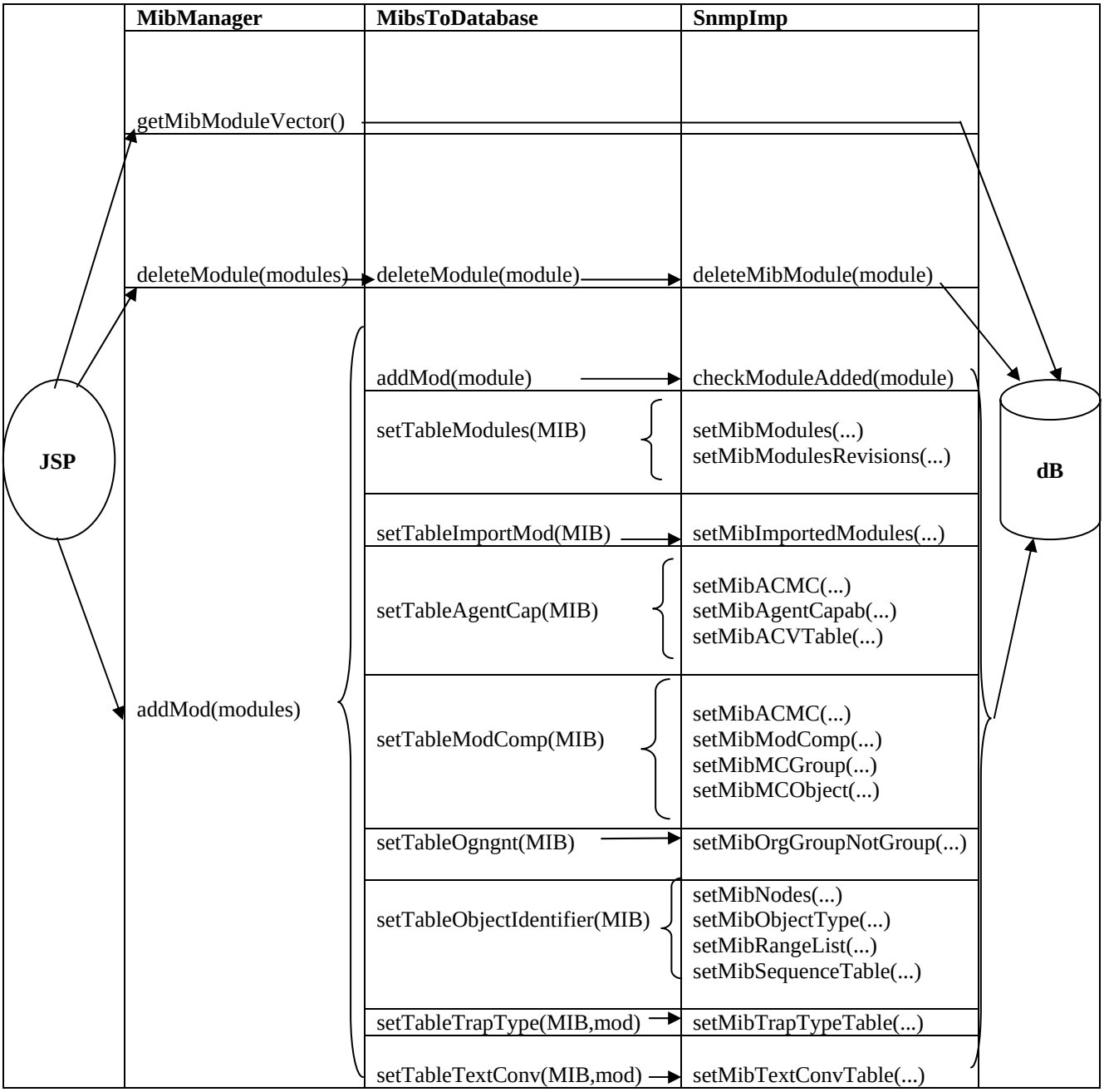

Figure 6.8 MibManager Classes

Where:

- *modules* is the modules's name that have to be deleted or added.
- *module* is a single module's name.
- *MIB* is the instance that is created when a module is loaded, it contains all the module information.
- (...) is all the information needed to add one file in the table.

6.5 MIB BROWSER

The MIB Browser is the application that manages the equipments. It shows the status of the equipments and it allows changing its configuration.

It can be described by three parts related to each other:

MIB Tree : it contains all the SNMP nodes from the modules stored in the database.

MIB Object Information: it displays the basic object information, such the name, oid, module where the object belongs and the enumeration or range in case that the object has it.

Query the equipment: this tool gives the possibility to query the equipment for the status of the different parameters and the possibility to change them if they are configurable.

6.5.1 Location in the GNMS file structure

The main Java modules are in the */gnms/gnc/command/special/mib*
The HTML and JSP modules are in */gnms/gnc/command/special/mib/jsp* directory.

6.5.2 Development

The figure below represents the MIB Browser structure and contains all the classes used:

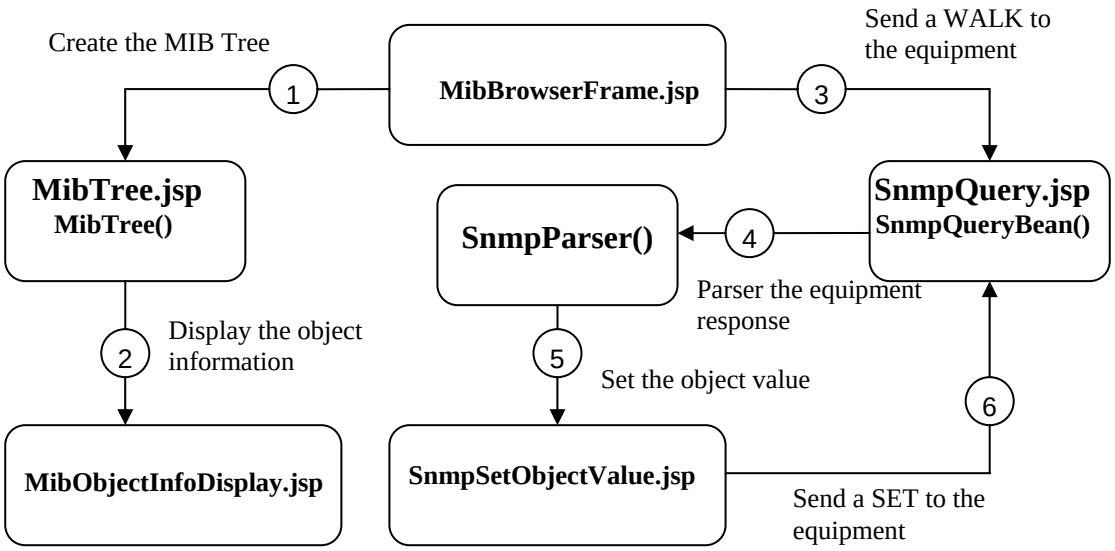

Figure 6.9 MibBrowser Structure

The MIB Browser performs three basic functions, on dependant of what we want to do, it will follow a way or another one:

- Display the object information (1,2)
- Query the equipment with a WALK(1,2,3,4)
- Change an object value by means of a SET (1,2,3,4,5,6)

Here's a list of all the classes and JSP:

MibObjectInfoDisplay: this is a JSP that connects to the database to get the information about the object checked.

MibTree: this JSP calls the MibTree class in order to display the MIB Tree.

SnmpQuery: this JSP executes the queries WALK and SET.

MibBrowserFrame: this JSP gets the basic equipment information (site, and equipment) and prepares the query.

SnmpSetObjectValue: this JSP gets the new value that will have the object, and return the command to do the SET query.

SnmpQueryBean: this Java Bean contains all the necessary information to execute the query (siteId, equipId, protocol, and command)

SnmpParser: this class gets the equipment response, it translates the response and sorts it in a XML.

SnmpImp: this class contains all the queries to the database.

In order to clarify the different MIB Browser parts, they will be explained separately.

6.5.3 MIB Tree

6.5.3.1 Development

The figure below represents the MIB Tree structure and contains all the classes used.

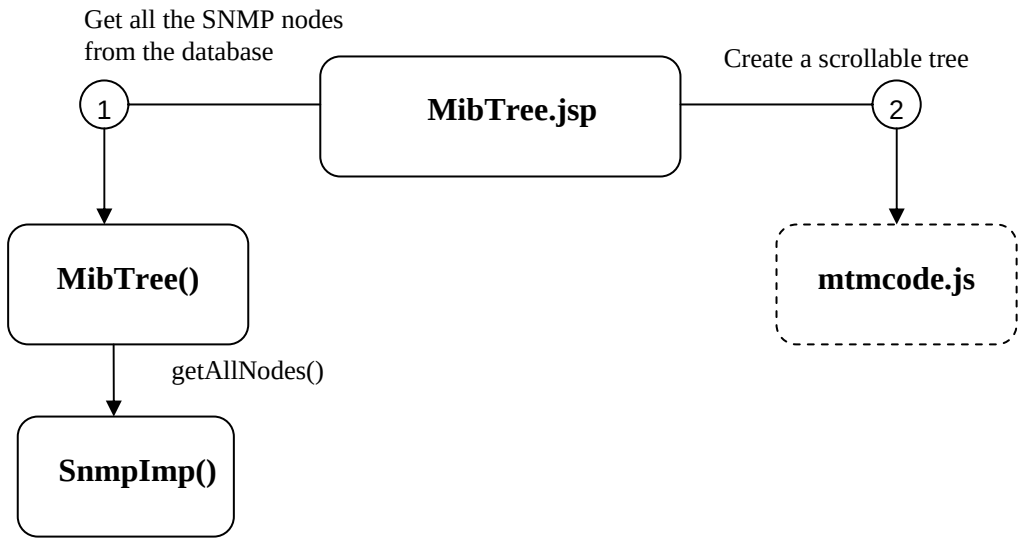

Figure 6.10 MibTree Structure

Each part develops the following tasks:

MibTree : this JSP is the responsible to create the MIB Tree and display it on the screen.

MibTree(): this class gets all the SNMP nodes and objects calling SnmpImp.

SnmpImp(): this class contains all the queries to the database, in this case it gets all the nodes and objects.

mtmcode : this is a JavaScript program that creates the structure and look of the Tree, this code was got it from www.treemenu.org , Morten's JavaScript Tree Menu. Some changes were done but the main structure was respected.

6.5.3.2 Database Queries

getAllNodes(): this query gets all the SNMP nodes present in the database. It is used for creating the MIB Tree.

6.5.3.3 Representation

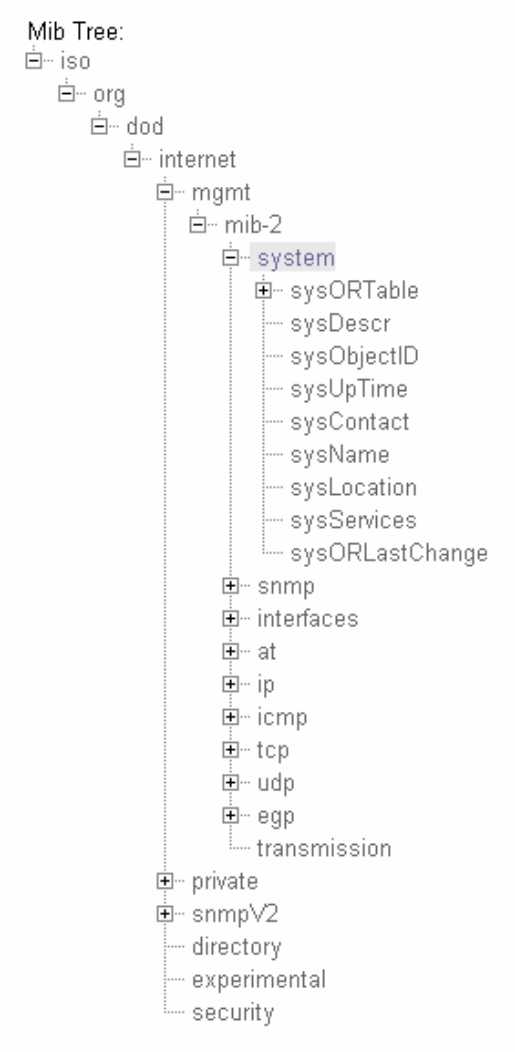

Figure 6.11: MIB Tree representation

The figure above is the MIB Tree representation, it contains all the SNMP nodes and objects stored in the database, as we can see the tree is scrollable, this fact contributes to help the easy and quick management of the objects.

When an object is clicked, the object information will appear on the MIB Browser main frame, this part will be detailed in the next section.

6.5.3.4 Code Structure

The MIB Tree code structure is quite simple, the MibTree JSP calls the MibTree() function to get all the nodes and objects, once they are got the MIB Tree is represented using the JavaScript mtmcode.

6.5.4 MIB Object Information

6.5.4.1 Development

The figure below represents the MIB Object Information structure and contains all the classes used:

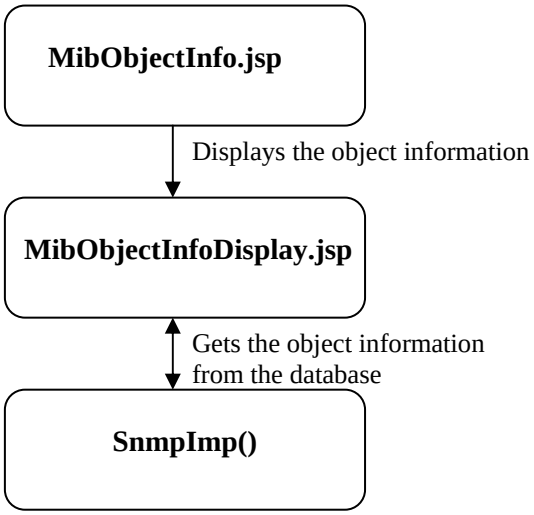

Figure 6.12: MIB Object Information Structure

Here's a list of all of them:

MibObjectInfo.jsp: it passes the oid to the top frame in order to keep it in case the equipment was queried, it calls the MibObjectInfoDisplay.jsp.

MibObjectInfoDisplay.jsp: displays the object information calling SnmpImp().

SnmpImp(): queries the database getting the object information.

6.5.4.2 Database Queries

getOIDInfo *(oid)* : gets the object information, such the name, description, the module where belong, and the enumeration or the range in case the object has it.

6.5.4.3 Representation

Mib Object Information:

Name	portBaudRate
Oid	1.3.6.1.4.1.594.4.1.10.4.1.1.6
Module	XSC-MIB
Description	"The baud rate of the connection."
Enumeration	0,nine-six;1,nineteen-two;2,twentyeight-eight;3,thirtyeight-four;4,fiftysix

Figure 6.13: MIB Object Information representation

This representation will appear on the main frame.

6.5.5 Code Structure

MibObjectInfo JSP Implementation

This code is quite simple, it calls the MibObjectInfoDisplay passing the oid and it passes too the OID to the parent frame.

MibObjectInfoDisplay JSP Implementation

Pseudo Code

```
//Import the libraries
//Initialize the variables

Request the oid parameter

Document oidDoc = call getOIDInfo(oid) from SnmpImp.java

Display the information using the xsl
```

6.5.6 Managing the equipment

6.5.6.1 Development

The figure below represents MibBrowserFrame structure and contains all the classes used:

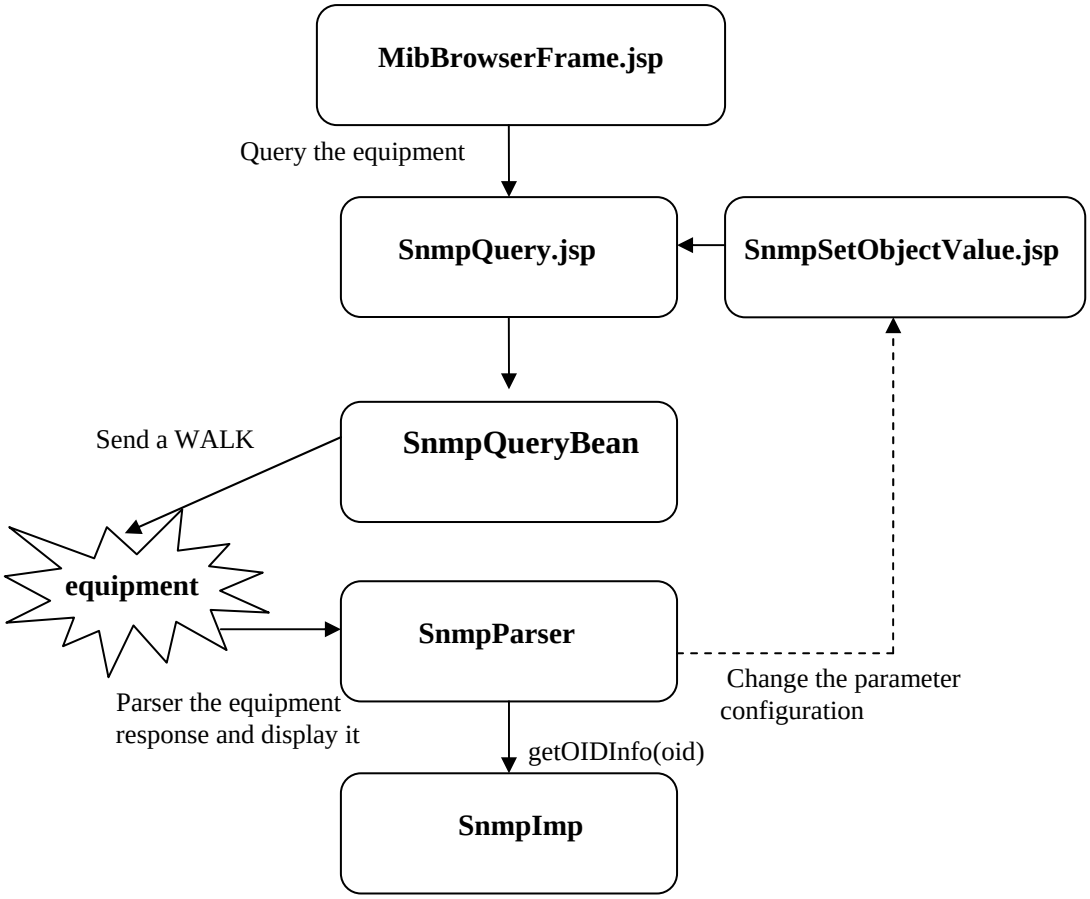

Figure 6.14: MibBrowserFrame Structure

Here's a list of all of them:

MibBrowserFrame.jsp: this JSP gets the equipment information (site, and equipment) and prepares the query.

SnmpQuery.jsp: this JSP executes the queries WALK and SET.

SnmpSetObjectValue.jsp: this JSP gets the new value that will have the object, and return the command to do the SET query.

SnmpQueryBean : this Java Bean contains all the information necessary to execute the query (siteId, equipId, protocol, and command)

SnmpParser : this class gets the equipment response, it translates the response and sorts it in a XML.

SnmpImp: this class contains all the queries to the database.

6.5.6.2 Database Queries

getEquipmentParameters *(siteIdInt,equipIdInt)*: this query gets the IP address and the equipment's name.

getOIDInfo *(oid)*: this query gets the object information (description, type, ...etc) by its OID

6.5.6.3 Representation

Scalar object

WALK

Default SNMP display format:

Object Id	Object Name	Object Type	Object Value
1.3.6.1.2.1.1.1.0	sysDescr	OCTET STRING	PEER Networks, a division of BMC Software, Inc., OptiMaster Release 2.3 on Lynx
1.3.6.1.2.1.1.2.0	sysObjectID	OBJECT ID	1.3.6.1.4.1.442.1.1.2.3.0
1.3.6.1.2.1.1.3.0	sysUpTime	TIMETICKS	1458704049
1.3.6.1.2.1.1.4.0	**sysContact**	OCTET STRING	Alan Clark
1.3.6.1.2.1.1.5.0	**sysName**	OCTET STRING	XSC
1.3.6.1.2.1.1.6.0	**sysLocation**	OCTET STRING	
1.3.6.1.2.1.1.7.0	sysServices	INTEGER	72

Figure 6.15: Walk a node representation

The figure above is the result of walking the tree by the “system” node, when the MIB Tree is walked by one node the application will show the status of all the objects that are under this node. We can also send a walk to a single object, and then the application will only show the status of the object in question.

As we can see there are some names of the objects in bold, these objects are the configurable ones, in this case we could change the parameters *syscontact*, *sysName* and *sysLocation*. Clicking in the bold name it will be displayed the “set page” that gives the possibility to send a SET to the equipment.

SET

OID	Name
1.3.6.1.2.1.1.4.0	sysContact

Set value: Lee Lilley

Set Cancel

Figure 6.16: SET representation

Clicking the *sysContact* this page will be displayed. The parameter *syscontact* can be changed pressing the Set button

Columnar object

WALK

Default SNMP display format:

Object Id	Object Name	Object Type
1.3.6.1.4.1.594.4.1.10.4.1.1.1.1	portNumberIndex	INTEGER
1.3.6.1.4.1.594.4.1.10.4.1.1.6.1	portBaudRate	INTEGER
1.3.6.1.4.1.594.4.1.10.4.1.1.7.1	portDataBits	INTEGER
1.3.6.1.4.1.594.4.1.10.4.1.1.8.1	portStopBits	INTEGER
1.3.6.1.4.1.594.4.1.10.4.1.1.9.1	portParity	INTEGER

portNumberIndex	portBaudRate	portDataBits	portStopBits	portParity
1	0 nine-six	8 eight	1 one	0 none
2	0 nine-six	8 eight	1 one	0 none
3	0 nine-six	8 eight	1 one	0 none
4	0 nine-six	8 eight	1 one	0 none

Figure 6.17: WALK a columnar object representation

In this case the “ports” node is walked, there are 5 columns, being one of them the index that allows recognizing the different ports.

Clicking the bold letters we can change the parameters.
SET

The set representation for a columnar object will be the same than in the scalar one, the only difference is that the number of the index will be passed to the “set page” in order to know which port will be changed.

6.5.6.4 Code Structure

MibBrowserFrame JSP Code implementation

Pseudo Code

```
//Initialize parameters

Request the siteId
Request the equipmentId

Document equipment = getEquipmentParameters(“siteId”,”equipId”)


//The IP Address and the equipment name will be used to display them
//on the browser

Get the IP Address from the document

Get the equipment name from the document

<html>
// MIB Browser main page

//The equipment name and the IP Address will be display on the top of
//the browser

<title>MIB Browser: <%=equipName%> - <%=ipAdress%></title>

<script>

function setSelectedOID(oid)
{
      Show the oid selected in the MIB Tree on the top frame (bottom1
      frame)

}

function snmpQuery(form)
{
      get the community name
      if(a WALK is sent)
            //Prepare the WALK command
            command = WALK command

      else if( a SET is sent)
            //Prepare the SET command
            command = SET command
}


</script>

// Create all the frames that will contain the MIB Browser

// The top frame will contain the buttons to send the query and to
//choose the community name implemented in the bottom1.html

<frame name="bottomFrame1" src="gnc/speccmd/mib/html/bottom1.html">

//This frame is used only to launch the MibTree on the menu frame,
the //size of this frame is 0

<frame name="code" src="gnc/speccmd/mib/MibTree.jsp?<%=parameters%>">

//Create a frame an split it in two frames
```

```
<frameset>

      // This frame will contain the MIB Tree, in the meanwhile it
      //will be showed a page saying that the MIB Tree is loading
      <frame name="menu" src="gnc/speccmd/mib/html/menu_empty.html">

      // This is the main frame where the object information is
      //displayed and where the user see the parameters status and
      //send the command SET. At the first moment the frame will be
      //empty.

      <frame name="mainFrame"
      src="gnc/speccmd/mib/html/bottom2.html">

</frameset>
</html>
```

Graphical Code

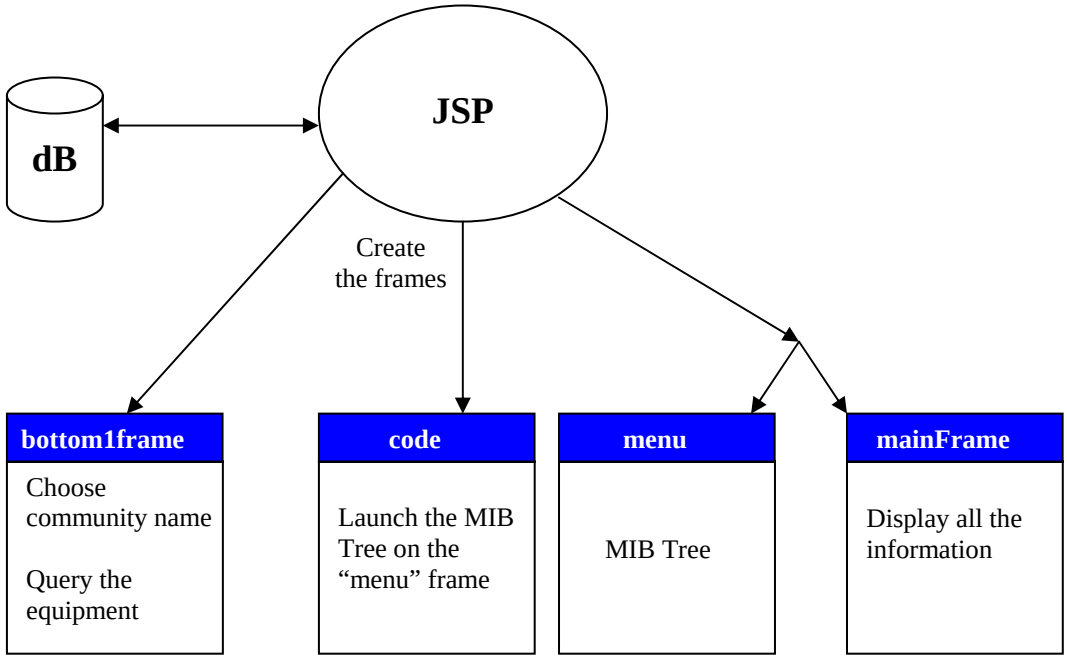

Figure 6.18: MibBrowserFrame Graphical Code

To clarify the position of the frames on the browser see the figure bellow:

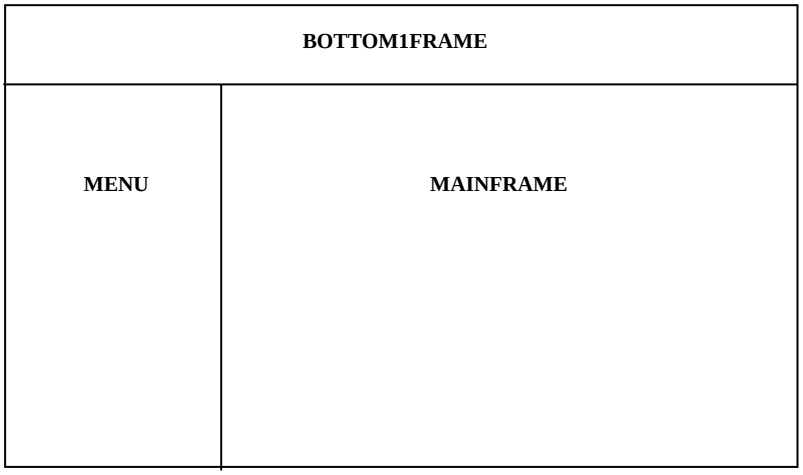

Figure 6.19: MIB Browser frames

As we can see, the code frame doesn't appear as this frame is used only to launch the MIB Tree.

SnmpQuery JSP Implementation Code

```
import libraries

use SnmpQueryBean bean

Initial declarations

// Request parameter 'command'
String command=request.getParameter("command");

// Execute the SNMP query with the bean
Document responseDoc = snmpBean.executeSnmpQuery();
```

SnmpSetObjectValue JSP Implementation code

```
import libraries
// Request parameters

oid = request.getParameter("oid");
name= request.getParameter("name");
type= request.getParameter("type");
index = request.getParameter("index");

<html>
    <head>
      <title>Snmp Set Object Value</title>
       <script>
            function prepareForm()
            {
                  //get the community name
                  //get the siteId
                  //get the equipId

                  if(it is an scalar object)
                        command =
                        SET:::community:::oid.0;;;value;;;type
```

```
                    //if it is a columnar object
                    else
                          command =

      SET:::community:::oid.index.0;;;value;;;type
            }

       </script>
</head>
<body>
      //create the form, on submit calls prepareform() and the
      SnmpQuery
      <form name="snmpForm" action="…/SnmpQuery.jsp" method="POST"
      onsubmit="prepareForm()">

      // create all the buttons and the inputs for the form

</body>
</html>
```

6.5.6.5 Parsing the equipment response

The SNMP Parser is the responsible for getting and parsing the equipment response. In the following figure the main structure is shown:

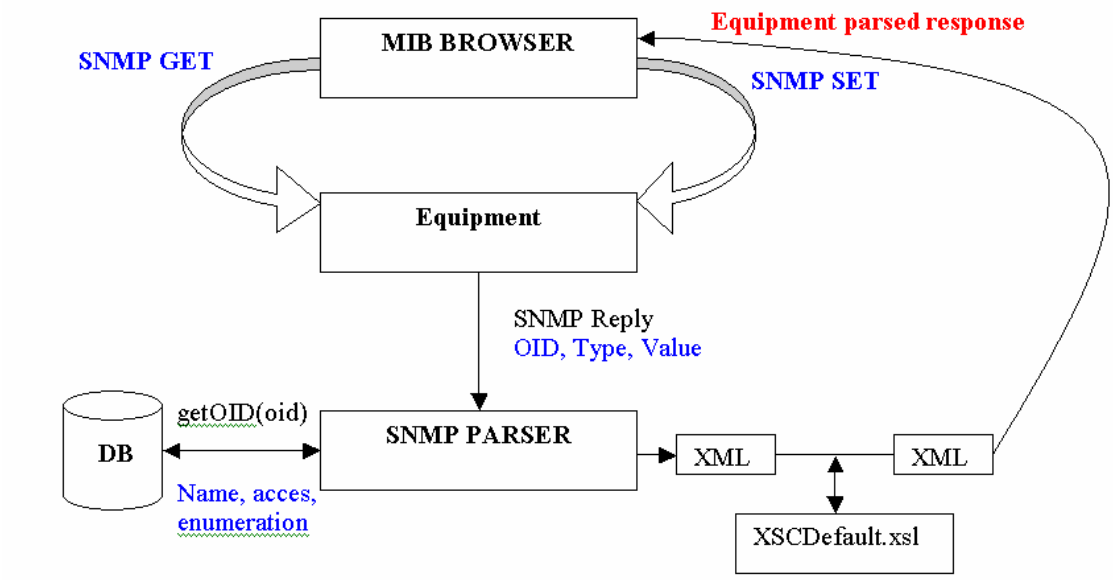

Figure 6.20: SNMP Parser Structure

When a SET is sent to the equipment, the object value will be changed and the equipment will response with a *true* or *false* on dependant if the operation has carried out with success.

When the equipment is queried by a WALK or GET, it will send the OID, the type and the value of the object/s queried, following the next response structure:

ObjId:::*objectIdentifier1***Type**:::*objectType1***Value**:::*objectValue1***ObjId**:::*objectIdentifier2*
Type:::*objectType2***Value**:::*objectValue2 …*

The SNMP Parser will query the database to get the name, the access and the enumeration (if it's a columnar object) of the object/s in question sending the *getOID* function, in order to be displayed to the user. The parser will construct an XML file with all the information collected:

<XscCommandResponse>
<Object table="false**">**
<OID> *objectIdentifier1***</OID>**
<Type> *objectType1***</Type>**
<Value> *objectValue1***</Value>**
<Name>object name 1**</Name>**
<**Access>**object access 1**</Access>**
</Object>
<Object table="false**">**
<OID> *objectIdentifier2***</OID>**
<Type> *objectType2***</Type>**
<Value> *objectValue2***</Value>**
<Name> object name 2**</Name>**
<Access> object access 2**</Access>**
</Object>
<Object >
…
</Object>
</XscCommandResponse>

This XML will be displayed in the MIB Browser using the XSCDefault.xsl style sheet.

In the next two chapters we will find the MIB Manager and the MIB Browser user's guide.

7. MIB MANAGER USER GUIDE

7.1 GUI

The GNMS GUI will have a consistent look for all pages. The screen will be divided into three main areas: the logo frame, menu frame and main frame. All business logic modules shall adhere to this design.

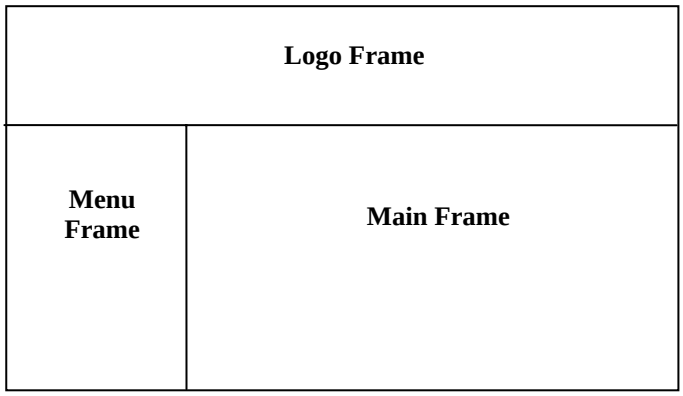

Figure 7.1: GNMS Page Layout

Logo Frame

The logo frame will contain nearly static information. The main feature will be the logo of GNMS, or for the business logic modules, their logo. The links to the right hand side logo frame will be links to other web pages / systems that are frequently used by GNMS users. This is currently set to be the BT Intranet home page etc. These links will bring up the linked page in a new window.

The navigation bar of the Logo Frame will change based on the page. It will contain a link to each of the previous pages in the web site tree.
E.g., if the user is in the Console page, this will show '**GNMS → Console'** where GNMS is a hyperlink to the GNMS home page.

Figure 7.2: Logo Frame Details

Menu Frame

The menu frame has a more dynamic content than the logo frame. For the GNMS home page, this frame will display links to the available business logic modules.

7.2 MIB Manager

The MIB Manager is accessed from the “Menu Frame” on the left hand side of the GNMS home page. First, we will follow the Console link.

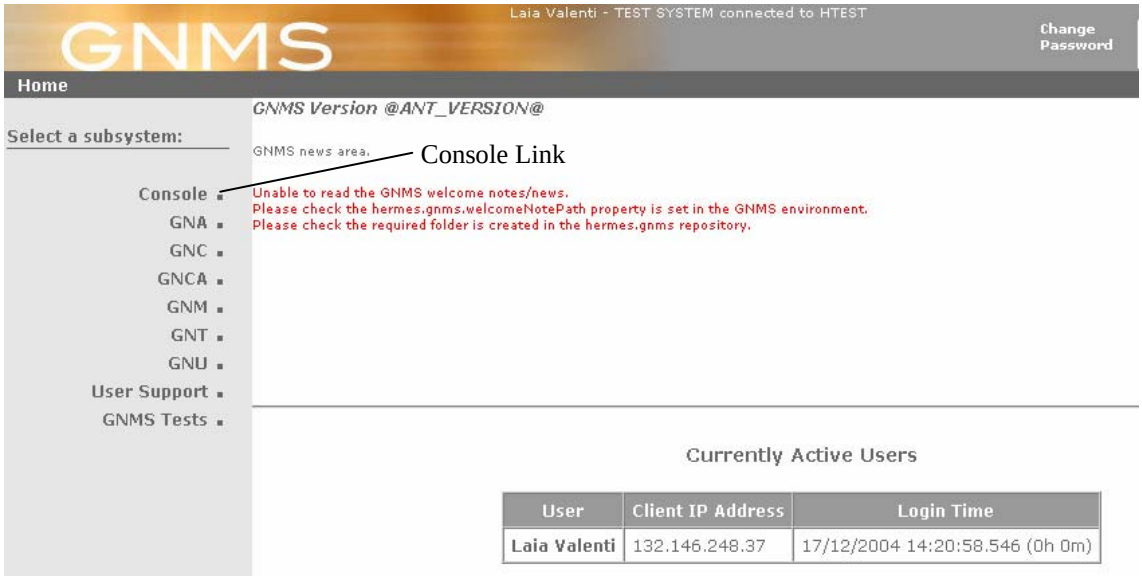

Figure 7.3: GNMS home page

Once we are in the Console subsystem, using the scrollable menu on the left and choosing the menu *Repositories* and *MIB Manager* under it, the main MIB Manager page is displayed:

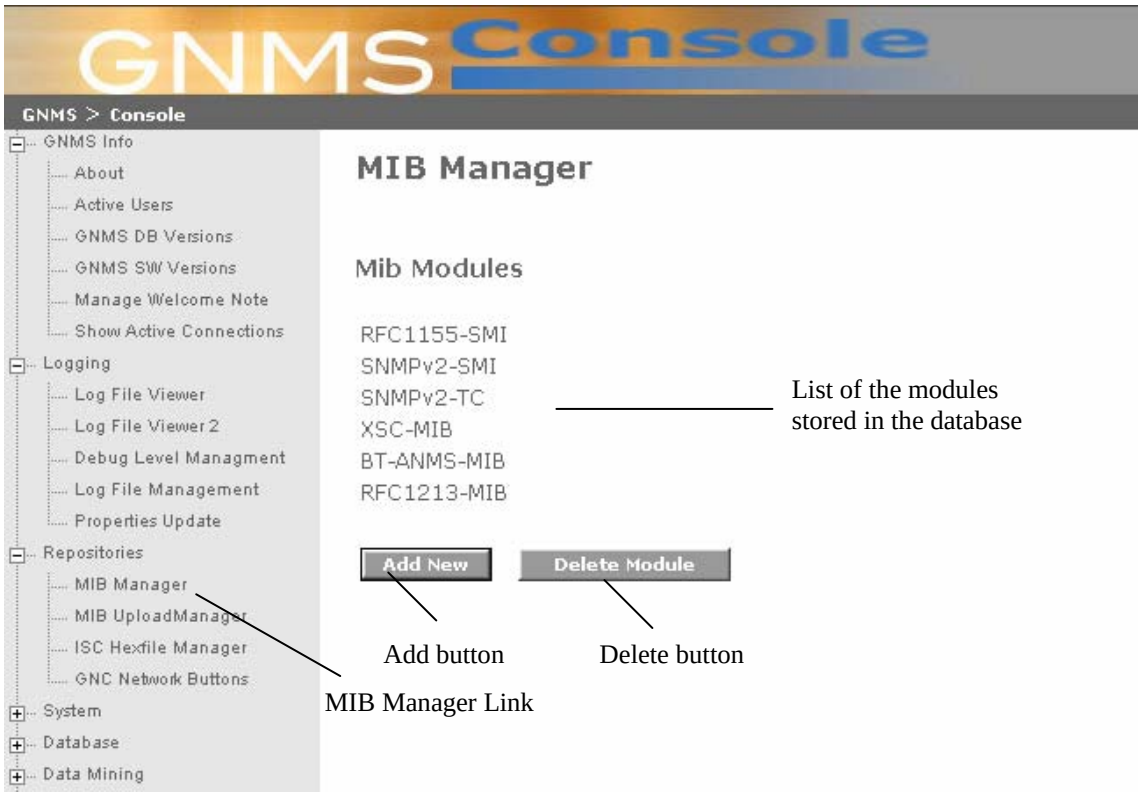

Figure 7.4: MIB Manager main page

In the MIB Manager main page, we can see the MIB modules already stored in the database. There are two buttons, one to add the modules and the other one to delete them.

Adding modules

Clicking on the button “Add New”, the module files stored in the repositories will be shown in a list box (further on it will be explained how to add the modules on the repositories)

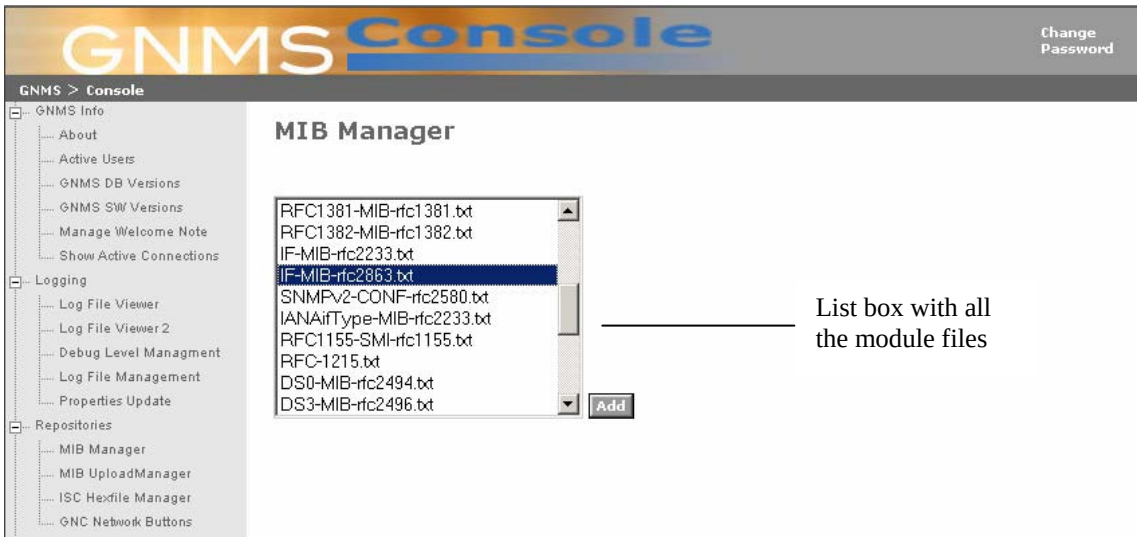

Figure 7.5: The add module page

We can choose one or more of them to be added in the database, if they are dependant of other modules it will display a message telling which modules must to be load before (see figure 7.6). If the modules are stored successfully, a success message will be shown (see figure 7.7)

Figure 7.6: Adding a module that is dependent on another one

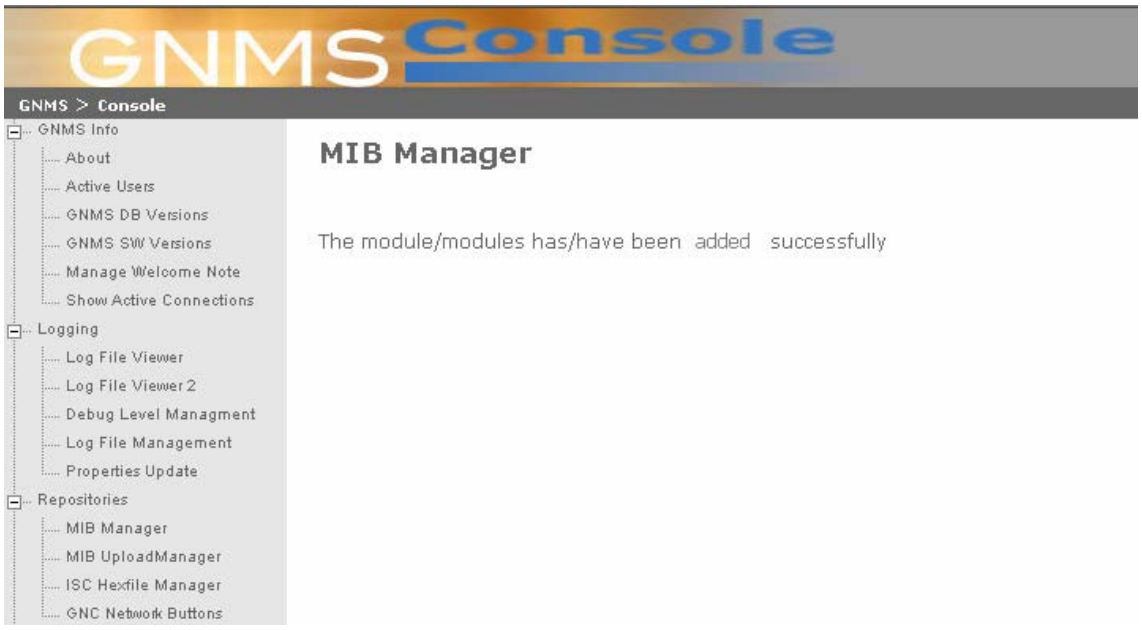

Figure 7.7: Success message

Deleting modules

Clicking the “Delete module” button on the MIB Manager main page (see figure 7.4), the modules stored in the database will be displayed in a listbox, the user can choose the module/s that have to be deleted.

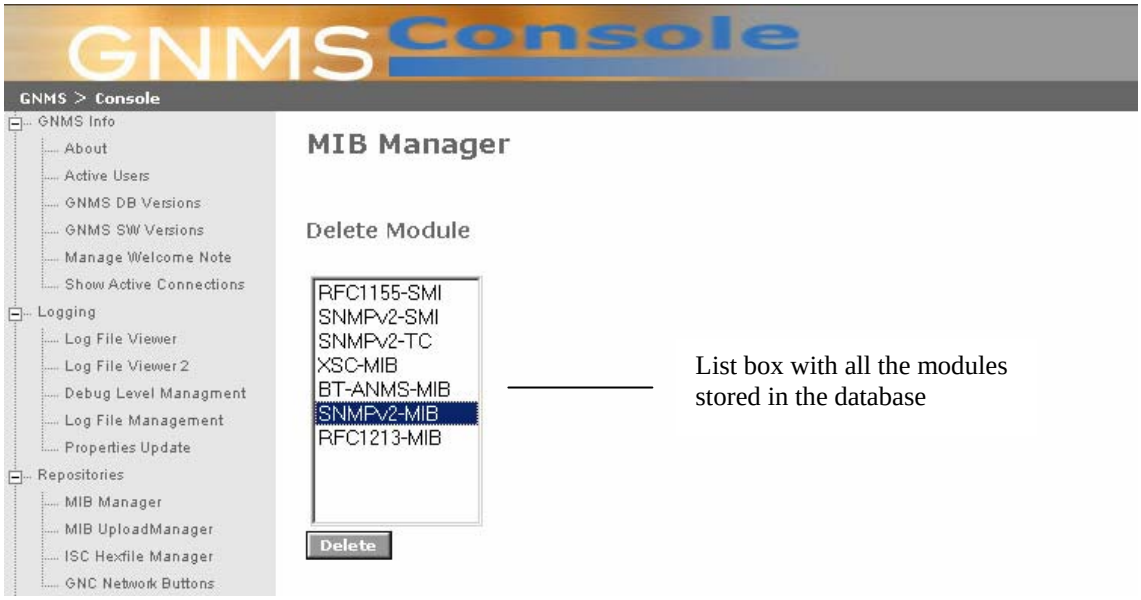

Figure 7.8: The delete module page

Once we delete the module/s successfully, we will see the following message:

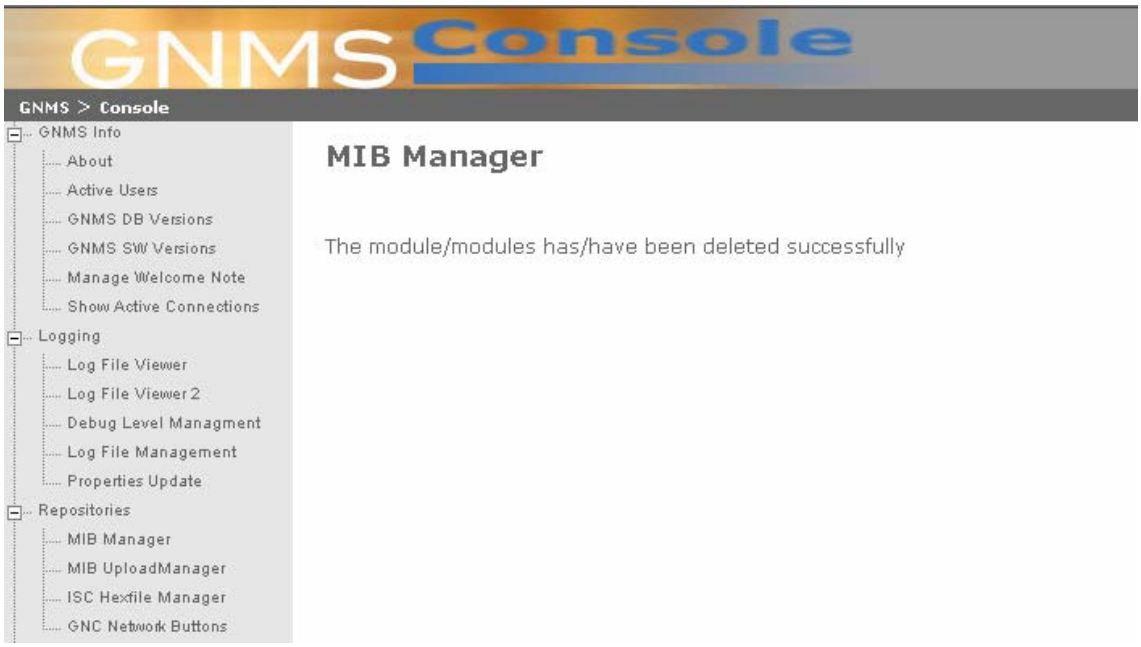

Figure 7.9: The delete success message

Managing the repositories

We will follow the *MIB UploadManager* link in the left hand menu to access into the repositories.

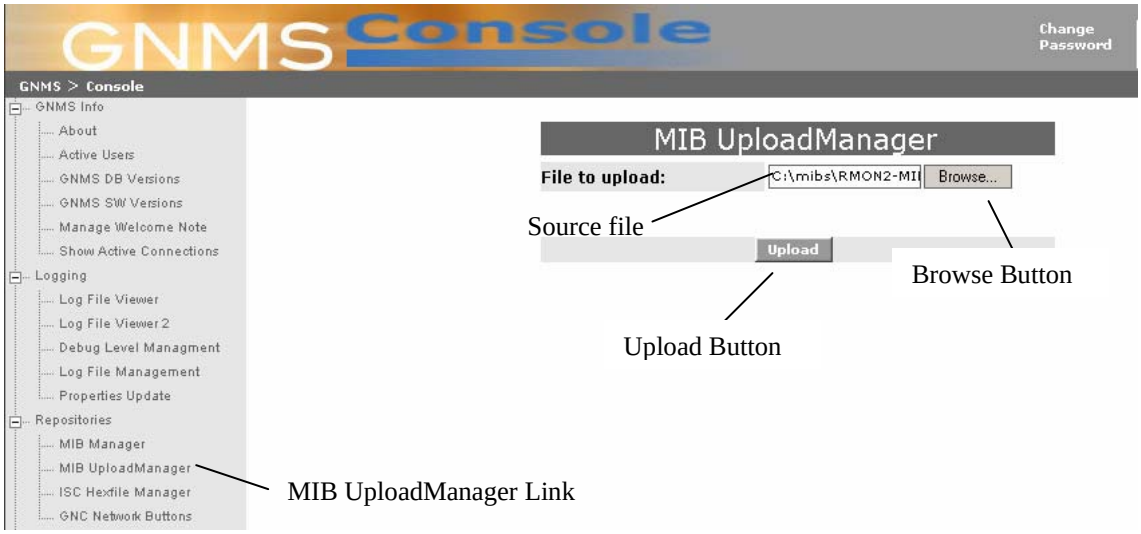

Figure 7.10: Upload Manager Page

We use the “Browse” button to select a file to upload, it will be stored in the repositories pressing the “Upload” Button.

If the file is well loaded, it will appear the following message:

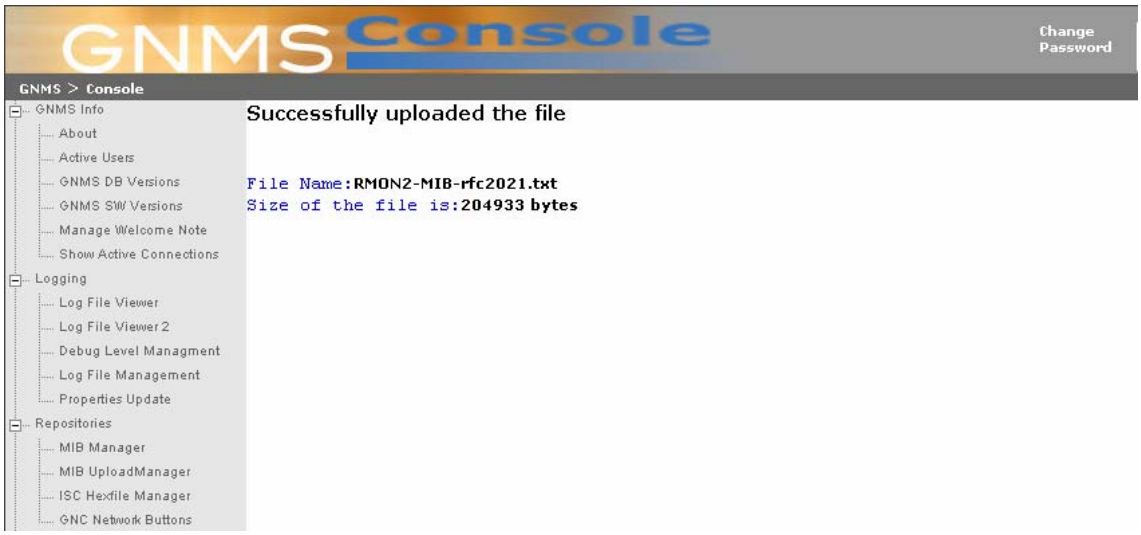

Figure 7.11: File uploaded message

Once we load a file in the repositories, we will be able to add it in the database using the MIB Manager.

8. MIB BROWSER USER GUIDE

The MIB Browser is a SNMP MIB Browser that allows loading MIB's, MIB Browsing, walking MIB tree and performing some of the SNMP related operations.

The MIB Browser:

- Provides the capability to load and view MIB Modules in a MIB Tree.
- Facilitates walking the MIB tree
- Facilitates performing of the Basic SNMP Operations.
- Queries the equipment by means of a SNMP WALK, showing the value of the object selected.

To access to the MIB Browser we log into GNMS, following the GNC link from the "Menu Frame".

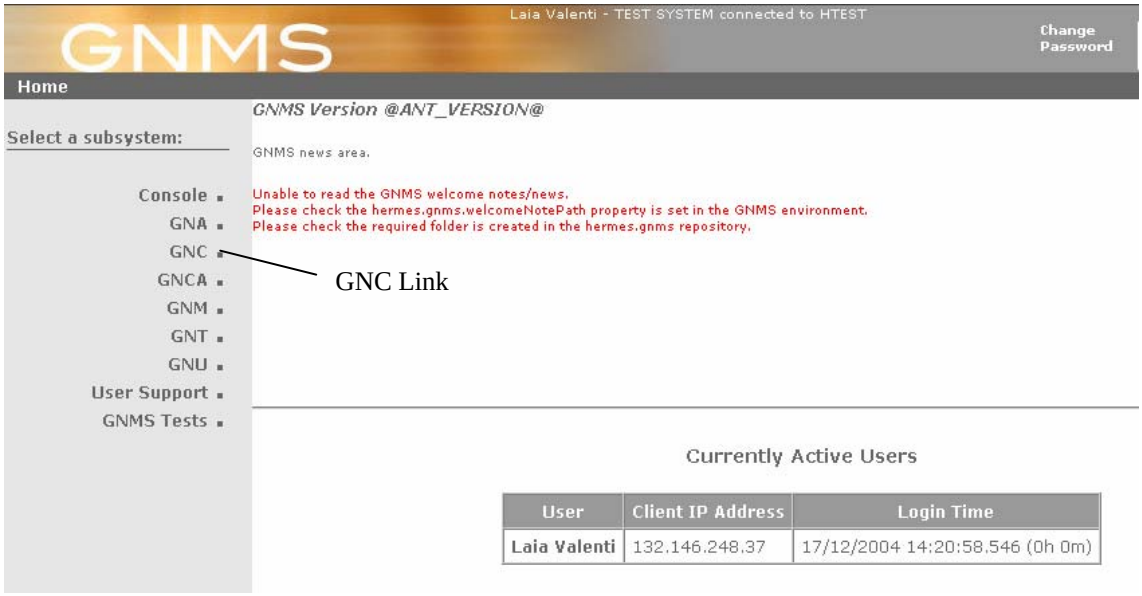

Figure 8.1: GNMS main page

Once we click on the GNC link, we enter into the GNC System.

Actually, there are two show stations that we can manage using the MIB Browser, XSC (eXtended Site Controller) and ANMS (AdVanced Network Management System).
The user can access to the equipment choosing the equipId and the siteId or using the search tool.

In the example we connect to XSC by using the search tool, the following page appears:

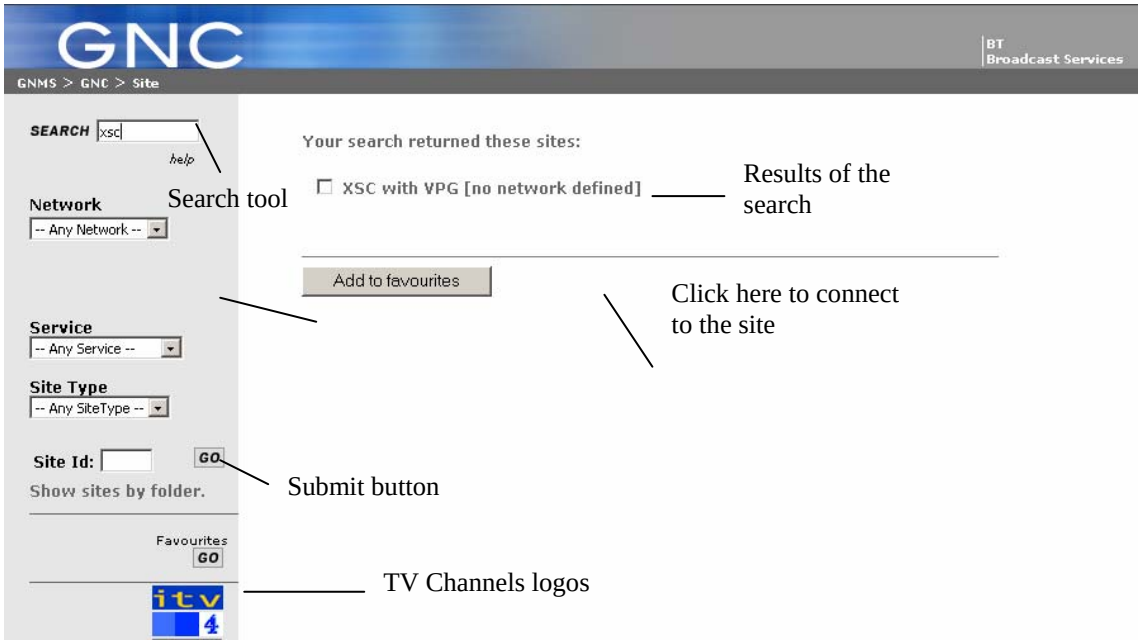

Figure 8.2: GNC system

By clicking on the specific result we connect to the site, we follow the MIB Browser link on the “main frame” to access to the MIB Browser:

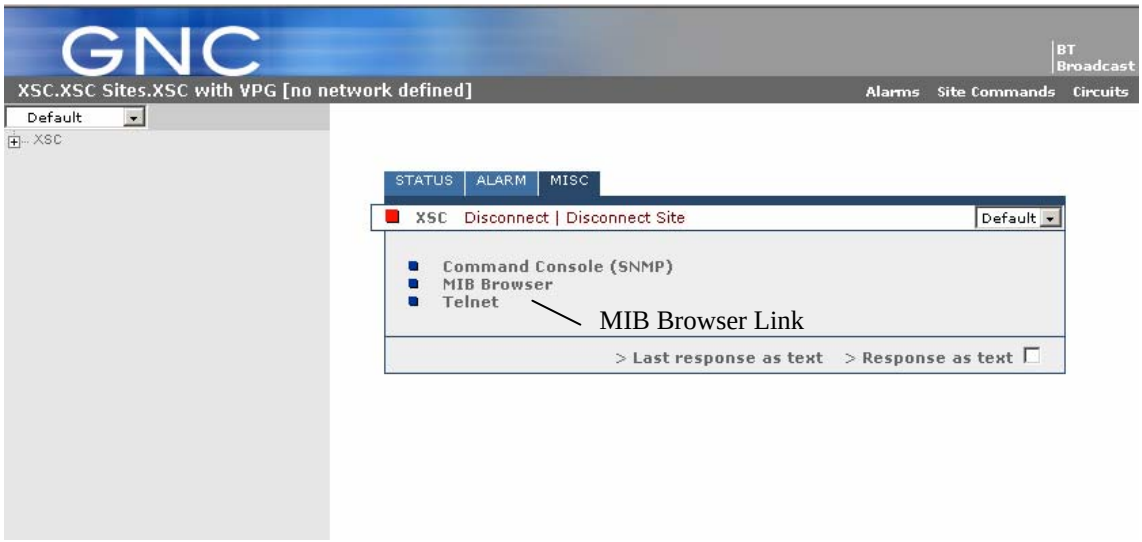

Figure 8.3: Connection to the XSC site.

Clicking on the MIB Browser link, a new window is opened:

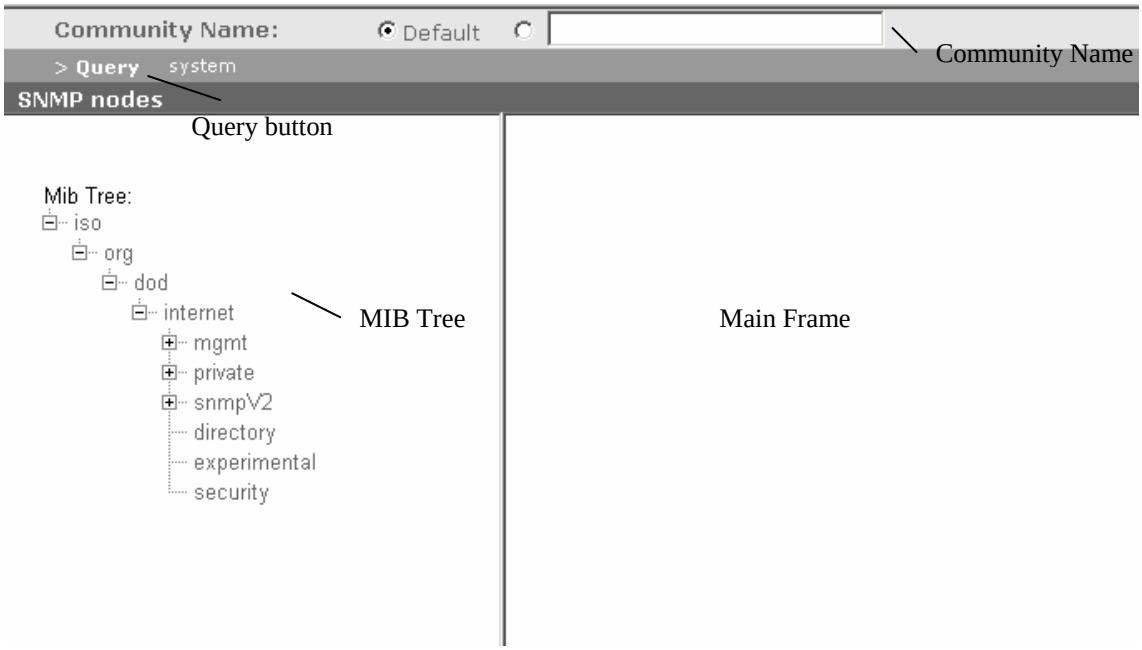

Figure 8.4 MIB Browser

SCALAR OBJECT

MIB Object Information

Using the MIB Tree, we can access to any object to see its information. The user clicks on the object name and the object's information is displayed on the Main Frame:

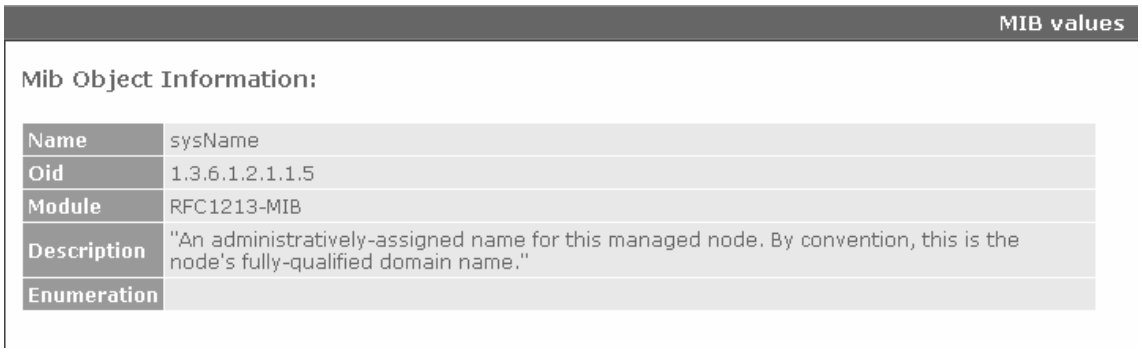

Figure 8.5 MIB Object Information

Walking and setting an object

When we click on an object on the MIB Tree, the object OID will be passed to the top frame, pressing the button "Query" a WALK is sent to the equipment.

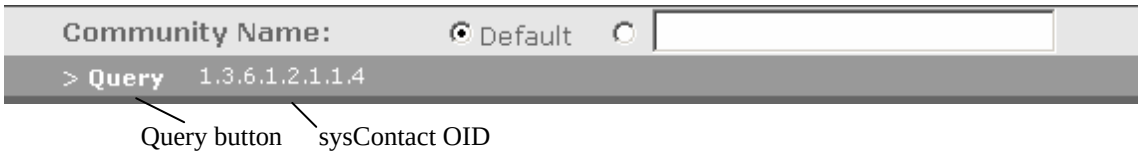

Figure 8.6: Query button

If we press the “Query” button on the Main Frame, the result of the query appears:

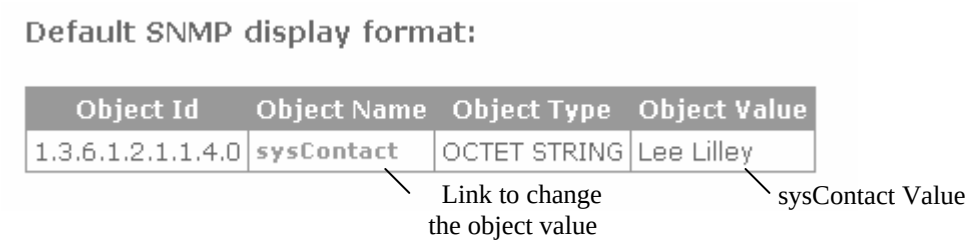

Figure 8.7 Equipment response

To change the object value, the user would click on the object name “sysContact”, the SET page will appear on the Main Frame. Pressing the “Set” button, the object value is changed, otherwise if we click on “Cancel”, we come back to the previous page without changing the value.

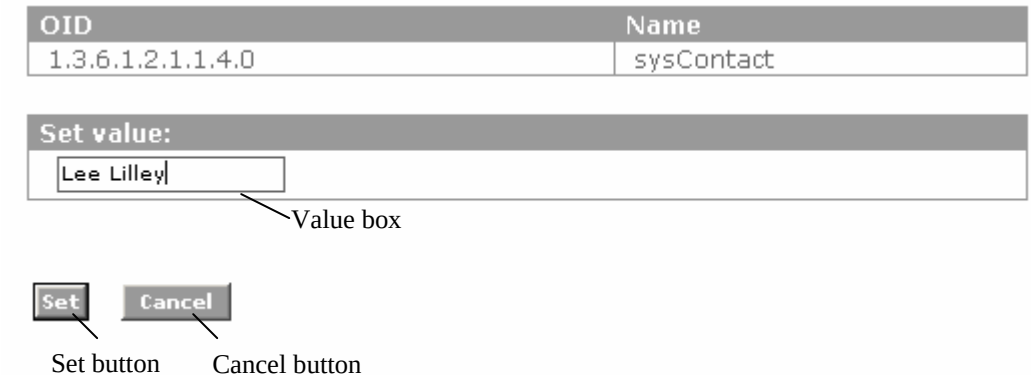

Figure 8.8 Set page

Walking a node

To walk an object is the same that send a GET to the equipment, the WALK is useful in order to walk a node. In this example we walk the node system, the result is the following:

Default SNMP display format:

Object Id	Object Name	Object Type	Object Value
1.3.6.1.2.1.1.1.0	sysDescr	OCTET STRING	PEER Networks, a division of BMC Software, Inc., OptiMaster Release 2.3 on Lynx
1.3.6.1.2.1.1.2.0	sysObjectID	OBJECT ID	1.3.6.1.4.1.442.1.1.2.3.0
1.3.6.1.2.1.1.3.0	sysUpTime	TIMETICKS	1458704049
1.3.6.1.2.1.1.4.0	sysContact	OCTET STRING	Alan Clark
1.3.6.1.2.1.1.5.0	sysName	OCTET STRING	XSC
1.3.6.1.2.1.1.6.0	sysLocation	OCTET STRING	
1.3.6.1.2.1.1.7.0	sysServices	INTEGER	72

Figure 8.9 Walking a node

The user can only change the objects that have their name in bold, in this case are *sysContact*, *sysName* and *sysLocation*. Clicking on them the set page respectively appears in the Main Frame (see figure 8.8)

COLUMNAR OBJECT

MIB Object Information

Clicking on a columnar object in the MIB Tree, the information is displayed in the same way than in a scalar object.

Mib Object Information:

Name	portBaudRate
Oid	1.3.6.1.4.1.594.4.1.10.4.1.1.6
Module	XSC-MIB
Description	"The baud rate of the connection."
Enumeration	0,nine-six;1,nineteen-two;2,twentyeight-eight;3,thirtyeight-four;4,fiftysix

Figure 8.10 MIB Object Information (columnar object)

Walking and setting a columnar object

When the user queries a columnar object, the equipment response is displayed in the following way:

Default SNMP display format:

Object Id	Object Name	Object Type
1.3.6.1.4.1.594.4.1.10.4.1.1.1.1	portNumberIndex	INTEGER
1.3.6.1.4.1.594.4.1.10.4.1.1.6.1	portBaudRate	INTEGER
1.3.6.1.4.1.594.4.1.10.4.1.1.7.1	portDataBits	INTEGER
1.3.6.1.4.1.594.4.1.10.4.1.1.8.1	portStopBits	INTEGER
1.3.6.1.4.1.594.4.1.10.4.1.1.9.1	portParity	INTEGER

portNumberIndex	portBaudRate	portDataBits	portStopBits	portParity
1	0 nine-six	8 eight	1 one	0 none
2	0 nine-six	8 eight	1 one	0 none
3	0 nine-six	8 eight	1 one	0 none
4	0 nine-six	8 eight	1 one	0 none

portParity value in the port 2

Figure 8.11 Walking a columnar object

In order to change the value of the objects the user needs to click on the value of the proper one, for example, if the user would like to change the *portParity* in the port 2 he would have to click in the place where the narrow of the figure is pointing.
Once we click on the object value, the set page appears, it is similar to the one for the scalar object.

9. VALIDATION

Once the project was finished, we perform a few tests in order to confirm that the application is working correctly and satisfy the requirements.

9.1 MIB MANAGER VALIDATION

Add Modules

Action	Validation Result
Add one module not dependant	The module is loaded and the confirmation message is displayed.
Add one module which is dependant on another one being already loaded	The module is loaded and the confirmation message is displayed.
Add one module which is dependant on another one that is not yet loaded	Display a message with the name of the module that have to be added.
Add one module which is dependant on another one that in his turn is dependant on another one.	Display a message with the name of the first module that have to be added.
Add 10 modules (dependant, not dependant,...)	The modules not dependant and the dependants of another one that are already loaded are stored. For the other cases the error message is shown

Delete Modules

Action	Validation Result
Delete one module that not has others modules dependent of it	The module is deleted and the confirmation message is displayed.
Delete one module that has others modules dependent of it	Displays a message saying that the module cannot be deleted because there are modules that are dependent of it
Delete a group of 10 modules where the both cases above are represented.	The modules that haven't others modules dependent of them are deleted. For the other case, the message is shown

9.2 MIB BROWSER VALIDATION

Action	Validation Result
Show the MIB tree, containing all the nodes and objects of the MIB modules loaded.	The tree is shown correctly
Expand/collapse the nodes	This functionality is working correctly
Access to an scalar object information by clicking the name on the MIB tree	The object information (Node Name, OID, Module and Description) is displayed on the main frame
Query an scalar object by pressing the “Query” button	The object type and the object value are displayed on the main frame
Press the “Set” button to change the object value	The object value is changed
Click the “Cancel” button to abort the operation	It returns to the equipment response.
Walk a node by clicking the “Query” button	Get the information of all the objects that the node has. If the objects are configurable it is possible to change their value.
Access to a columnar object information by clicking the name on the MIB tree	The object information (Node Name, OID, Module, Description and Enumeration) is displayed on the main frame.
Query a columnar object	The object type and the object values are displayed on the main frame.
Walk a node that contains columnar objects	Displays two tables, one with the OID and the object types, and the other one with the values of the columnar objects.
Change the value of the columnar object	The object value is changed

10. CONCLUSIONS

This thesis presents the new SNMP MIB Browser that is integrated into the Global Network Management System (GNMS) application, which is used to manage and control BT Broadcast Services Terrestrial TV network.

The SNMP MIB Browser application can be logically divided into three modules:

- **MIB Manager**: this web-based application manages the MIB Modules, allowing the user to load and delete the MIBs from database tables.

- **MIB Browser**: this web-based application manages and controls the equipments using the SNMP. It allows to control and manage the equipments by sending WALK and SET messages, which query the equipment for the MIB variables that are hierarchical displayed in a tree.

- **Database**: is a relational database that stores the MIB variables

Before the development there were some important decisions that we had to make in order to improve the final result of the whole application. Here is a list of some of them:

- **Requirements**: that was the first step of my project. I wrote the “State of Requirements” document for the new module. This document contains the functional and non functional requirements talking about the:

    - Graphical User Interface.
    - Database Services.
    - Specific requirements of the three main modules.
    - Design.
    - Performance.

  Chapter 4 contains the initial agreed requirements included in the “GNMS Circuit Tracker State of Requirements” document, which are 100% covered in the final development.

- **Understand the MIB Modules Files**: I had to familiarize myself with the MIB Modules structure and the information contained in them before to design the database. The section 3.4 contains the syntax of the MIB Modules.

- **Database design :** Once I understood the MIBs, I prepared to design the database that would contain the MIBs information. Mapping the MIB objects to a relational database was a difficult task, the MIB information model consists of lots of interrelated objects. In the section 6.3 we can find the database structure.

- **Java SNMP Library research:** Before to start the application development, we had to find an additional suitable Java Library in order to speed up and make the information extraction from the MIBs easier. The section 6.2 details which libraries were analyzed and the procedure adopted for determining which one was the best for our task. Finally we choose the AdventNet SNMP Library, the principal reason was the time spent loading the MIB Modules. The AdventNet SNMP Library features are detailed in the section 5.4.

- **Servlet or JSP**: we can talk about SNMP MIB Browser application as a server-side application. Then, after that for every program that I was about to write I had to decide between using proper servlets or JSPs. In chapter 4 there is a comparative between both technologies.

The SNMP MIB Browser was implanted and integrated into the GNMS, it worked correctly, with all the asked requirements and functionalities accomplishes.

A User Guide exists for both applications and summarises the way that these applications work. These User Guides are in the Chapter 7 and 8.

GLOSSARY OF ACRONYMS AND ABBREVIATIONS

ANMS	AdVanced Network Management System
ANSI	American National Standards Institute
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract System Notation One
B&SC	Broadcast and Satellite Communications
BER	Basic Encoding Rules
BGP	Border Gateway Protocol
BS	Broadcast Services
BT	British Telecom
CCITT	Comité Consultative Internationale de Telegraphique et Telephonique
CISCO	Computer Information System Company
CMIP	Common Management Information Protocol
CMIS	Common Management Information
CMOT	CMIS and the CMIP protocol with a TCP/IP transport
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
DCA	Data Configuration Application
DNS	Domain Name System
DVB	Digital Video Broadcast
EGP	Exterior Gateway Protocol
EJB	Enterprise Java Bean
EOF	End of file
ERP	Enterprise Resource Planning
FDDI	Fiber Distributed Data Interface
FTP	File Transfer Protocol

GIF	Graphic Interchange Format
GNA	Global Network Alarm (Acquisition System)
GNC	Global Network Control (and Configuration Application)
GNdB	Global Network Database
GNM	Global Network Monitoring (Application)
GNMS	Global Network Management System
GNU	Global Network User (Management Application)
GUI	Graphical User Interface
HPOV	Hewlet Packard OpenView
HTML	Hyper Text Mark-up Language
IAB	Internet Architecture Board
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IESG	Internet Engineering Steering Group
IETF	Internet Engineering Task Force
IIOP	Inter-ORB Protocol
IP	Internet Protocol
IRSG	Internet Research Steering Group
IRTF	Internet Research Task Force
ISC	Intelligent Site Controller
ISO	International Organization for Standardization
ITU	International Telecommunication Union
ITU-T	Telecommunication Standardization Sector
J2EE	Java 2 Enterprise Edition
JAR	Java Archive Files
JASMI	Java API for SMI

JDBC	Java DataBase Connectivity
JDK	Java Development Kit
JFC	Java Foundation Classes
JMS	Java Messaging Services
JPEG	Joint Photograph Experts Group
JSP	Java Server Pages
JTA	Java Transaction API
JTS	Java Transaction Service
JVM	Java Virtual Machine
LAN	Local Area Network
LMs	Layers Managers
MCFG	Main Configuration Database
MDI	Multiple Document Interface
MIB	Management Information Base
MIBs	MIB Module Files
NMS	Network Management System
OID	Object Identifier
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
OSS	Operational Support System
PC	Personal Computer
PDU	Protocol Data Unit
PL	Programming Language
QoS	Good Quality of Service
RFC	Request For Comment
RIP	Routing Information Protocol
RMI	Remote Method Invocation
SDK	Software Development Kit

SGML	Standard Generalized Markup Language
SGMP	Simple Gateway Monitoring Protocol
SMAEs	Systems Management Application Entities
SMI	Structure of Management Information
SNMP	Simple Network Management System
SQL	Structured Query Language
SQLJ	Structured Query Language in Java
SSL	Server provides Secure Sockets Layer
TCP	Transmission Control Protocol
TLV	Tag, length, and value
TMN	Telecommunications Management Network
TOAD	Tool for Oracle Application Developers
UDP	User Datagram Protocol
UK	United Kingdom
UML	Unified Modelling Language
UPC	Universitat Politècnica de Catalunya
UTC	Coordinated Universal Time
WAP	Wireless Access Protocol
WG	Working Group
XML	eXtensible Mark-up Language
XSC	eXtended Site Controller
XSL	eXtensible Style-sheet Language: Transformations
XSLT	eXtensible Style-sheet Language: Transformations