

**SYMANTEC**

**Project Title: LEAF-SNMP**

**PROJECT SYNOPSIS**

## CONTEXT

A computer network consists of network elements like nodes, links, routers etc. The network administrator (manager) is the one who manages the network. Administrator faces problems in managing the network.

Adding a new network element requires configuration maintenance. When the network is functioning some of the network elements like nodes, links, routers may go down. It creates obstacles in network functions. Also, various nodes use the network for variable times; hence for accounting the information about the use of network by every node must be maintained. The administrator or some special user must be able to find out performance of a node. If data in the network is confidential then it must be protected from outside world. Also there are different categories defined for the user and the data should be accessible to them as per their privilege level.

## PROBLEM

Various types of networks are existing with various topologies. Complete information about network (network devices and links) is to be maintained for network management.

Existing SNMP agent works for kernel version 1.0 (for Linux). It does not provide facility of extending MIBs (Management Information Base). If we need to enhance product for newly invented devices we can not enhance it. SNMP agent does not provide cross compatibility (Windows and UNIX).

SNMP agent for LEAF is not available. It creates security threats for network.

SNMP agent is supposed to manage information about network devices and links. It should store this information in Management Information Bases. SNMP agent is supposed to send traps on failure in network being monitored and should dispatch requested network information to SNMP manager.

System is expected to report for fault, configuration, accounting, performance, and security. It is also expected to provide GUI which allows administrator to set network parameters.

## SOLUTION

An SNMP (Simple Network Management Protocol) agent is an application that performs the operational role of receiving and processing requests, sending responses to the manager, and sending traps when an event occurs. Network administrator (manager) can perform various functions like checking whether elements are functioning properly or not. If a certain important device is not working or an important setting is breached, a trap can be set which alerts the administrator and he can take further action.

The project deals with implementing an SNMP agent covering all the features mentioned in RFC 1213 (defines how MIBs can be used with management protocols) and RFC 2863 (defines part of MIBs for network management use). The respective modules will be used to configure the variables specified in MIBs and provide the values configured when the variables specified in the MIBs and provide the values configured when queried.

LEAF (Linux Embedded Appliance Firewall) is a secure, feature rich, customizable embedded Linux network appliance for use in a variety of network topologies. LEAF is a project derived from the original Linux Router Project (LRP). LRP described a way to implement a customized Linux distribution which was small enough to fit on a floppy disk, and yet provide advanced routing functionality such as traffic shaping, fire-walling and even DHCP and DNS service. Both LEAF and LRP are designed to be as small as possible, primarily for reliability and security reasons. The small footprint of LEAF/LRP makes it ideal for recycling old PC hardware for use as routers. There are a number of different LEAF and LRP sub-distributions, each with a different focus or objective. For instance, the LEAF-Bering distribution (on which this paper is based) was the first LEAF distribution to use the Linux 2.4 kernel, in order to take advantage of the Shore wall firewall and other enhancements. Some of these features of LEAF would be emulated in our project.

The GUI helps the administrator to configure and manage the network device and makes the interaction simple and intuitive using dialogue boxes or menus rather than typing commands. MIB's provide a standard database to the network related scenario maintaining a common platform everywhere.

This SNMP agent provides

1. facility to enhance MIBs (Management Information Base)
2. Cross compatibility.
3. User Interface.
4. Network management for kernel version 2.6 onwards.

Thus the task of network management is made simpler using SNMP agent which the project is going to extend.

**Benefits:**

- SNMP agents are present for kernel version 1.0 and less. This agent will work for kernel version 2.6 and above.
- This SNMP agent provides a user interface.
- It provides facility to extend MIBs as per requirements.

**SYMANTEC**

**Project Title: LEAF-SNMP**

# **FEASIBILITY STUDY REPORT**

## INTRODUCTION

The project involves building a SNMP Agent which works in a LEAF emulated Linux environment for configuration and retrieval of data through MIB's implemented for RFC 1213 and RFC 2863. It also involves developing a GUI for the SNMP Agent which facilitates for easier and efficient way of using the different tools to gather information about complex network of managed nodes.

## PURPOSE

The purpose of feasibility study is to analyze the existing process of network analysis and whether use of LEAF with SNMP and a GUI facility can help to enhance the network analysis and monitoring process. This document studies whether the proposed system is realizable or not. Also, it includes whether the problem exists and solution to be implemented would be able to solve the problem or not. It is also useful for deciding the constraints on the system and assumptions to be made while implementing the solution.

The feasibility study takes into account the following factors:

- Technology
- Hardware Characteristics
- Resources
- Network Complexity

The enhancement to the existing process of network analysis would simply make the analysis simpler by using the GUI to retrieve useful management, control information without having the need to remember a set of commands.

## METHODOLOGY

Following methodology was used to perform the feasibility study

1. Surveying: - This technique involves studying of the proposed system by finding out information from different sources related to problem area, alternative methods for implementation the same problem, etc...

A general search on what a SNMP Agent is? What are its features? What is LEAF? is done to get basic idea about the different entities involved in the proposed system along with their relationships.

An extensive search on how the SNMP Agent works, the different versions of SNMP, the role of SNMP manager, extension of MIB's is done with the help of different forums, mailing lists, sites to get a clear understanding of their usage.

A search on the configuration of LEAF and net-snmp toolkit which is used for testing purpose is made for setting up the above mentioned packages.

## REFERENCES

1. **SNMP, SNMPv2, SNMPv3, and RMON 1 and 2 3rd Edition**  
William Stallings  
Addison-Wesley, 1998
2. **LEAF Documentation**  
Bering-uClibC Team, March 2007
3. **Net-SNMP Documentation & Wiki**  
Net-SNMP Team, March 2007  
<http://en.wikipedia.org/wiki/Net-SNMP>
4. **CISCO PIX Firewall**  
[http://en.wikipedia.org/wiki/Cisco\\_PIX](http://en.wikipedia.org/wiki/Cisco_PIX)
5. **Net-SNMP and LEAF Mailing lists**

## GENERAL INFORMATION

### CURRENT SYSTEMS AND PROCESSES

In Industry, the combination of SNMP+LEAF is used for building management systems for firewalls. Also, LEAF can be used with some other protocols for building management systems for firewall appliances.

1. Cisco has implemented the management systems in its PIX firewall. There are two ways to get SNMP information from PIX, first is to query PIX using SNMP. The host will send query to PIX & gets response (polling). Second way is to have the PIX send the traps to the SNMP manager
2. Lucent has its own standard (TL1 – Transaction Language 1) for communicating with devices for management and configuration. CCITT recommends TMN which typically uses CMIP for management and configuration.

These methods have thus been implemented by the firms for their own use. The lack of GUI and the implementation only restricted to the Linux platform makes it difficult for the normal users interested in network analysis to use these systems. Also, the network administrator has to remember loads of shell commands which can be quite cumbersome.

Thus, the introduction of GUI in these management systems would make the retrieval of information a mouse click away. Also, some statistics might be depicted graphically which might be useful in analysis as an image speaks more than a text of thousand words.

## SYSTEM OBJECTIVES

The proposed system will provide a GUI for the network analysis which would help the network administrators to analyze and manage the complex network of the network devices in a more efficient way rather than issuing a set of commands to retrieve the same information. The system involves:

- Implementation of SNMP Agent which works in a LEAF emulated Linux environment for analyzing and managing complex network of the network devices.
- Reference of MIB's as per the RFC 1213 and RFC 2863 for extension of MIB's.
- Acting as a base model for future developments.
- Enhancing the existing process of retrieval of important network management information.

The end product expected will comprise of a GUI which allows the user to interact with the various network elements. The system is expected to be developed within a period of six months.

## ISSUES

The issues involved in the development of this system are:

- The MIB's implemented by the system should be restricted to a subset of the specifications as specified in RFC 1213 and RFC 2863.
- The utility must be such that it is backward compatible with the previous versions of the Linux Kernel.

## ASSUMPTIONS AND CONSTRAINTS

- The project is constrained to the implementation of Management Information Bases (MIB) as per specifications specified in RFC 1213 and RFC 2863 only.
- The project assumes that the appropriate version of Linux Embedded Appliance Firewall (LEAF) is emulated on the Linux Distribution.
- The project involves implementing the SNMP Agent on the Linux Platform. The cross-platform functionality will be addressed in the next version.
- The source code of this project would be available under the GNU General Public License

## **ALTERNATIVES**

The alternative to the proposed system is to continue with the existing management systems built for firewalls and firewall appliances which use a set of commands to query the firewall for the required network information. This would comparatively be more cumbersome due to the large number of commands involved and also the resultant information would be in a highly disorganized manner occurring in the console shell. The problem with this is that the network administrator has to keep track of all the commands issued and the outputs obtained during each query thus making the analysis process pretty tedious. Also, a new person would require loads of training in order to get acquainted with the system.

## **RECOMMENDATIONS AND CONCLUSION**

The primary objective of the project is to enhance the network monitoring and analysis process by providing a GUI which is easy to interact. This involves developing a SNMP Agent which would interact with the underlying LEAF which can act as a firewall, gateway and a router and supply the required information to the SNMP Agent. The SNMP Manager can then query the SNMP Agent for the required information which can be used for hardware failure detection, network management.

The alternatives to the above project are not able to optimally satisfy the requirements and in some cases no alternatives are present. Due to the above reasons, this project is highly recommended and would actually enhance the SNMP agent and make the network management process a bit simpler

**SYMANTEC**

**Project Title: LEAF-SNMP**

# **SOFTWARE PROJECT PLAN**

## OVERVIEW

- The motivation for this project is the need to implement an SNMP Agent with a GUI i.e. EMS(Elements Management System) which would help the network administrators to monitor a complex network without having them to remember the commands for querying for management information.
- The customer is network administrator and people involved in monitoring a complex network consisting of many nodes which are to be managed.
- The project will deliver a SNMP Agent along with the MIB's implemented by referencing RFC 1213 and RFC 2863 and providing a GUI for using the various features of the SNMP Agent.
- It will free of cost being an open source project. The source code will be available for modification and further enhancements.
- The first fully functional version will be released around 31<sup>st</sup> Jan, 2008. Future Versions and bug fixes would be released in the future.
- The organizations involved are Symantec Corp. Ltd and Vishwakarma Institute of Technology (VIT)

## GOALS AND SCOPE

### PROJECT GOALS

Project Goal	Priority	Comment/Description/Reference
<b>Functional Goals:</b>		
Functional goal #1		Manage network information.
Functional goal #2		Monitor Network
<b>Technological Goals:</b>		
Technical goal #1		Trace communication link
<b>Constraints:</b>		
Environmental		Restricted to Linux
Application specific standards		Following RFC-1213, RFC-2863

## **PROJECT SCOPE**

The project will deliver a SNMP Agent along with the MIB's implemented by referencing RFC 1213 and RFC 2863 and providing a GUI for using the various features of the SNMP Agent.

The project will not deliver cross platform functionality and compatibility with SNMPv1. Also the MIB's except RFC 1213 and RFC 2863 wouldn't be provided.

### **Included**

The project will deliver a SNMP Agent along with the MIB's implemented by referencing RFC 1213 and RFC 2863 and providing a GUI for using the various features of the SNMP Agent.

The receivers of this project would be Symantec Corp. Ltd and Vishwakarma Institute of Technology (VIT) .

### **Excluded**

This project will exclude.....

- Cross Platform Functionality
- Compatibility with SNMPv1
- MIB's implemented for other RFC's except RFC 1213 and RFC 2863

## SCHEDULE AND MILESTONES

Milestones	Description	Milestone Criteria	Planned Date
M0	Start Project	Budget Release	2007-08-25
	e.g.: Project goals and scope defined	PRS or SRS reviewed Stakeholders identified Impl. Proposal reviewed	2007-09-10
M1	Start Planning		2007-09-10
	<milestone description, e.g. Life Cycle Objectives LCO defined>	Scope and concept described	2007-09-15
M2	Project synopsis and System requirement specification.		
		Requirements agreed, project plan reviewed, resources committed	2007-09-22
	System Requirements Specification document	System Requirements Document reviewed	2007-09-22
M3	Use Case Diagrams	Use cases are verified.	2007-09-29
M4	Sequence Diagrams	Sequence diagrams are verified.	2007-09-29
M5	State chart Diagrams	State chart Diagrams are verified.	2007-10-06
M6	Activity diagrams	Activity diagrams are verified	2007-10-06

## DELIVERABLES

Identifier	Deliverable	Planned Date	Receiver
D1	SNMP Agent	April 2008	Symantec
D2	MIB	April 2008	Symantec
D3	GUI	April 2008	Symantec

**SYMANTEC**

Project Title: LEAF-SNMP

# SYSTEM REQUIREMENT SPECIFICATION

## INTRODUCTION

This document specifies functional requirements of system. It defines problem statement, product perspective and product position statement. SRS defines benefit of system and compares system to be developed with existing system and its drawbacks. Software requirement specification clearly gives hierarchy of functional requirements of system. This hierarchy is used to develop further use cases (cases that system can handle).

## PURPOSE

SRS is used to describe functional requirements of system in hierarchical manner. Purpose of SRS is to give hierarchy between goals and objectives of system to be developed. Further this document is used as base document to develop Analysis diagrams.

## SCOPE

The goal of this work is to produce an SNMP agent which works in a LEAF (Linux Embedded Appliance Firewall) emulated Linux environment. This will be managed by any SNMP manager for network management.

This product will work for network management in form of

1. Fault
2. Configuration
3. Accounting
4. Performance
5. Security

This will provide facility to extend MIBs which acts as information base for above tasks

### Benefits:

- Currently SNMP agents are present for kernel version 1.0 and less. This product will be working for kernel version 2.6 and above.
- This product provides user interface.
- It provides facility to extend MIBs as per requirements.

## DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

SNMP : Simple Network Management Protocol.  
LEAF : Linux Embedded Appliance Firewall.  
RFC : Request for comment  
SGMP : Simple Gateway Management Protocol

Term or Acronym	Definition
MIBs	Management Information Bases
	Fault, Configuration, Accounting, Performance and

## REFERENCES

Book Title : SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, Third Edition  
Author : William Stallings.  
Publisher : PEARSON Education.  
Book Title : Computer Networks.  
Author : Andrew Tanenbaum.

## OVERALL DESCRIPTION

### PROBLEM STATEMENT

The problem of	Network management
Affects	SNMP manager and user.
The impact of which is	Inconsistent information about network ultimately increasing cost.
A successful solution would	Provide an SNMP agent which will be managed by SNMP manager for network management (FCAPS).

### PRODUCT PERSPECTIVE

Product works in a LEAF emulated Linux Environment. It refers to RFC 1213 and 2863 which gives format of management information base. MIBs store all information about current status of network.

#### Block Diagram:

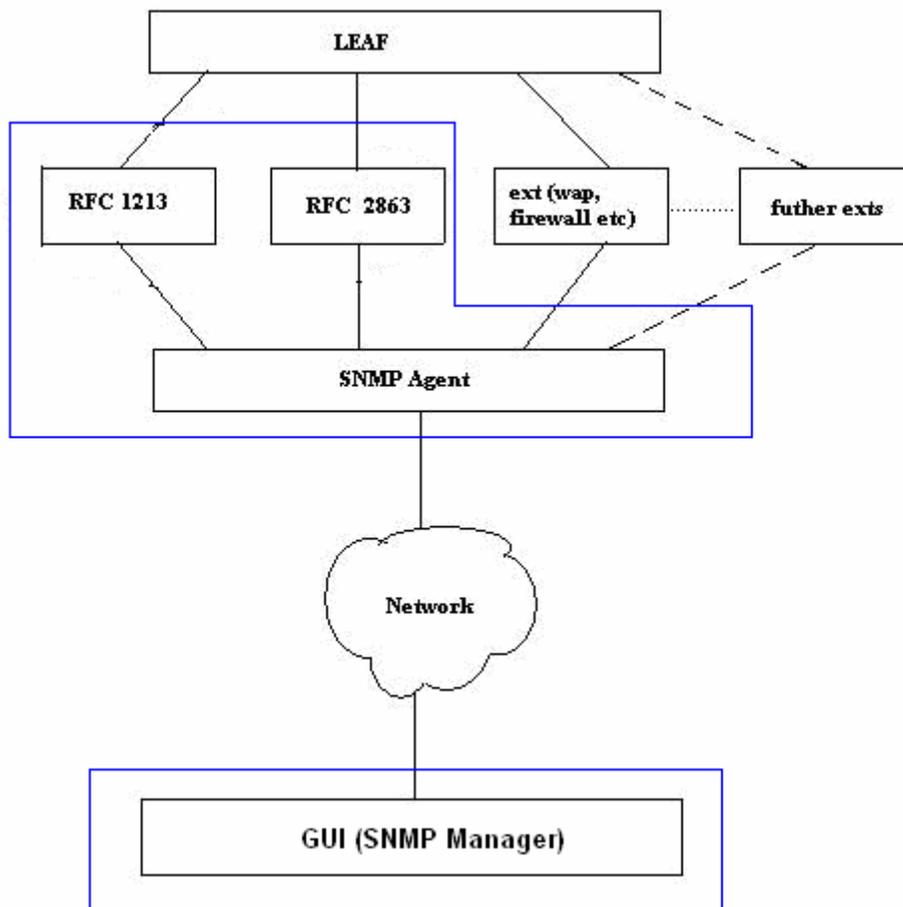


Fig 4.1 Block Diagram of the SNMP Agent in the LEAF emulated Linux Environment

## PRODUCT POSITION STATEMENT

For	SNMP manager and user
Who	Are network management people
The (product name)	SNMP agent
That	<ol style="list-style-type: none"><li>1. works for kernel version 2.6 and above</li><li>2. is managed by any SNMP manager</li><li>3. provides FCAPS</li><li>4. works on Linux environment</li></ol>
Unlike	Older versions of kernel (1.0 and less)
Our product	Works for current kernel version (2.6) and provides GUI for SNMP manager

### User Interfaces

GUI will display all information about polling (i.e. SNMP manager queries SNMP agent) and trapping (i.e. SNMP agent sends alerts to SNMP manager).

### Hardware Interfaces

NIC cards have 48 bit data link layer address. This address is registered when NIC card is inserted in device. Device that accounts to resources for nodes available in existing network is included in network information with complete information.

### Software Interfaces

SNMP manger (e.g. IBP Tivoli)

SNMP manager polls local SNMP agent with get request PDU to obtain device information and receives traps from SNMP agent. SNMP manager takes appropriate action (e.g. rebooting on software failure) and sends status to local SNMP agent.

### Communications Interfaces

SNMP – Simple Network Management Protocol.

SNMP is an application layer protocol. It is an enhanced version of SGMP. A network cannot be put together by human effort alone. The complexity of such system dictates the use of automated network management tools. As the networked installation become larger more complex and more heterogeneous cost of network management rises. The simple network management protocol has been developed to provide a tool for multi-vendor, interoperable network management. SNMP refers to standards for network management, including a protocol, a database structure specification and a set of data objects.

SNMP provides fault management, accounting management, configuration management, performance management, security management. For SNMP, the MIB is a database structure in form of a tree.

SNMP works around get, set and trap to manipulate and convey network information across the entire network for optimal network management.

## CONSTRAINTS

If shared resources increase than limited amount that agent can maintain, system will reboot.

## ASSUMPTIONS AND DEPENDENCIES

It is assumed that shared resources will not exceed capacity of agent.

## SPECIFIC REQUIREMENTS

### Functional Requirements Hierarchy

Goal: SNMP Agent should *monitor network devices* for network management.

Objectives:

- Maintain network transaction includes fault monitoring to maintain network channel information and report faults.
- Access policy includes providing proper node access and deciding access policies for non-registered SNMP managers.

Goal: SNMP Agent should *manage network information* for network management.

Objectives:

- Retrieve network information includes investigating network information and manipulating MIBs.
- Dispatch network information includes reply to manger polling and traps to manager on faults.

Goal: SNMP Agent should *manage Network \_MIB* for network management.

Objectives:

- Extend Management\_Info\_base includes defining objects according to ASN and registering those objects.
- Generate MIB\_Code includes generating C and header files to be modified as per requirement and embedding the MIB in the agent.

## LOGICAL DATABASE REQUIREMENTS

Type of information:

There are two type of MIBs Central MIB (residing on manager) and local MIB (residing on agent).

Management information base has various tables, object variables which contain information about network devices and links. These variables and attributes of tables are modified according to status of network. The information in bases is updated by SNMP agent and sent to manager on request.

Frequency of use:

MIBs are accessed by agents and manager as and when required by the SNMP requests.

## SOFTWARE SYSTEM ATTRIBUTES

### Reliability

SNMP agent monitors network and sends faults detected to managers. SNMP protocol provides data integrity resulting into secured and well managed network information.

### Security

Security includes

1. Linux Embedded Appliance Firewall: SNMP agent is to be implemented for LEAF emulated environment in Linux which traces every incoming and outgoing message. It provides indirect security from malicious attacks or unauthorized retrieval of data from system.
2. Session maintenance.
3. Access policies for access rights.

### Functional Hierarchy

Goal: Monitor network devices.

Objectives:

- Maintain network transaction.
- Maintain Access policy.

Goal: Manage network information.

Objectives:

- Retrieve network information.
- Dispatch network information.

Goal: Manage Network\_MIB.

Objectives:

- Extend Management\_Info\_Base.
- Generate MIB\_Code.

**SYMANTEC**

**LEAF-SNMP**

## **USE CASE DIAGRAMS**

## 1. USE CASE TEMPLATE

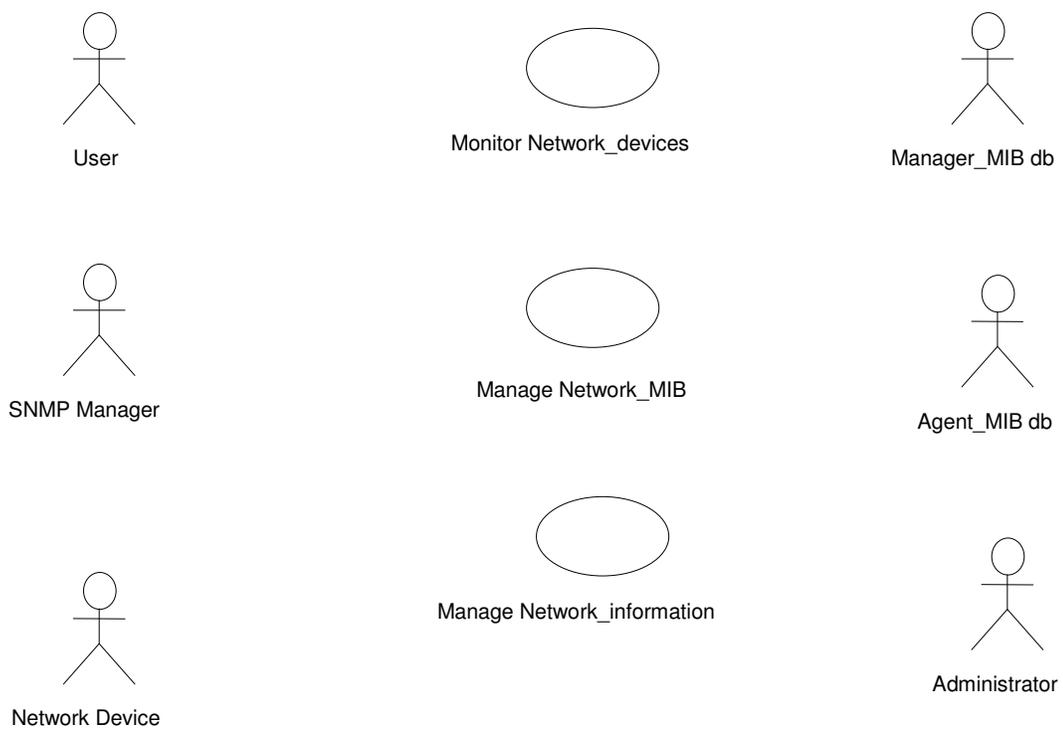


Figure 5.1 System Context Diagram

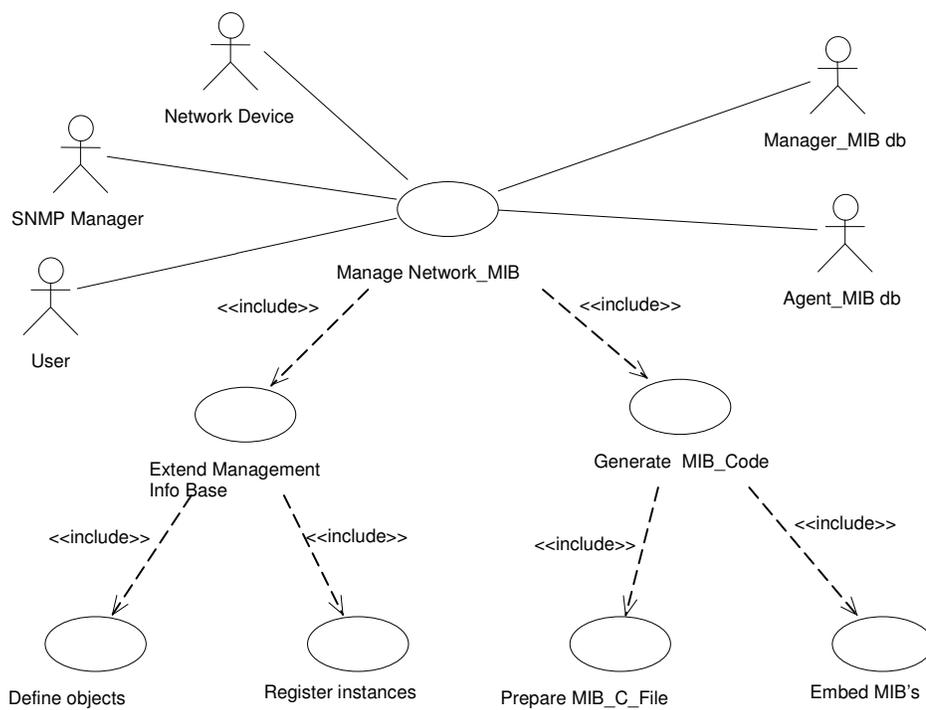


Figure 5.2 Requirement Capture Diagram for Manage Network\_MIB

<b>USE CASE #</b>	Manage network MIB	
Purpose	Managing network MIB deals with extension of MIB and code generation. Extension of MIB deals with defining new objects and registering instances of objects. Code Generation deals with generating the MIB C and header files.	
Preconditions	A new device pops-up for which MIB is necessary.	
Post conditions	MIB is generated and loaded to manager	
Primary Actors	1.SNMP manager 2.Administrator 3.Network device	
Trigger	SNMP manger asking for network information.	
<b>DESCRIPTION</b>	<b>Step</b>	<b>Basic Course of Action</b>
	1	Define object in the MIB according to ASN.
	2	Store the MIB file in the mibs directory.
	3	Generate C code for the MIB.
	4	Embed the MIB in the agent.
<b>DESCRIPTION</b>	<b>Step</b>	<b>Alternate Course of Action</b>
	1	Create and run SNMP daemon.
	2	Share object to be loaded dynamically.
	3	Change SNMP daemon configuration file and restart daemon.
<b>DESCRIPTION</b>	<b>Step</b>	<b>Error Scenario</b>
	1	MIB cannot be loaded.
	2	Wrong MIB definition conflicting with ASN.
	3	Segmentation Fault occurred.
	4	No instance of object at the OID.

<b>USE CASE #</b>	Extend Management_Info_Base.	
<b>Goal</b>	Extend MIB.	
<b>Purpose</b>	This use case will provide details about how a management information base (MIB) is extended.	
<b>Preconditions</b>	A new device pops-up for which MIB is necessary	
<b>Success Condition</b>	MIB is loaded.	
<b>Failed Condition</b>	MIB cannot be loaded and no instance of object.	
<b>Post conditions</b>	MIBs are extended.	
<b>Primary Actors</b>	1.SNMP manager 2.Administrator 3.Network device	
<b>Trigger</b>	Device is requesting for registration.	
<b>DESCRIPTION</b>	<b>Step</b>	<b>Basic Course of Action</b>
	1	Device sends request for registration
	2	Define objects in the MIB according to ASN.
	3	Store the MIB file in the mibs directory.
	4	MIB completes the registration procedure.
	5	MIB is loaded.
<b>DESCRIPTION</b>	<b>Step</b>	<b>Alternate Course of Action</b>
	1	Create and run SNMP daemon
	2	Share object to be loaded dynamically.
	3	Change SNMP daemon configuration file and restart daemon.
<b>DESCRIPTION</b>	<b>Step</b>	<b>Error Scenario</b>
	1	MIB cannot be loaded.
	2	Object cannot be found.

<b>USE CASE #</b>	Generate MIB_Code	
Goal	Generate the code for the MIBs	
Purpose	This use case will generate the code for the MIBs.	
Preconditions	Configuration file is specified	
Post conditions	The MIB code files are generated.	
Primary Actors	1.SNMP manager 2.Administrator 3.Network device	
Trigger	Mib2c is executed	
<b>DESCRIPTION</b>	<b>Step</b>	<b>Basic Course of Action</b>
	1	Get proper object ID (OID).
	2	Decide configuration file (e.g.min2c.scalar.conf).
	3	Give OID to mib2c.
	4	Execute mib2c.
<b>DESCRIPTION</b>	<b>Step</b>	
	1	Invalid OID.
	2	Private MIB cannot be loaded.

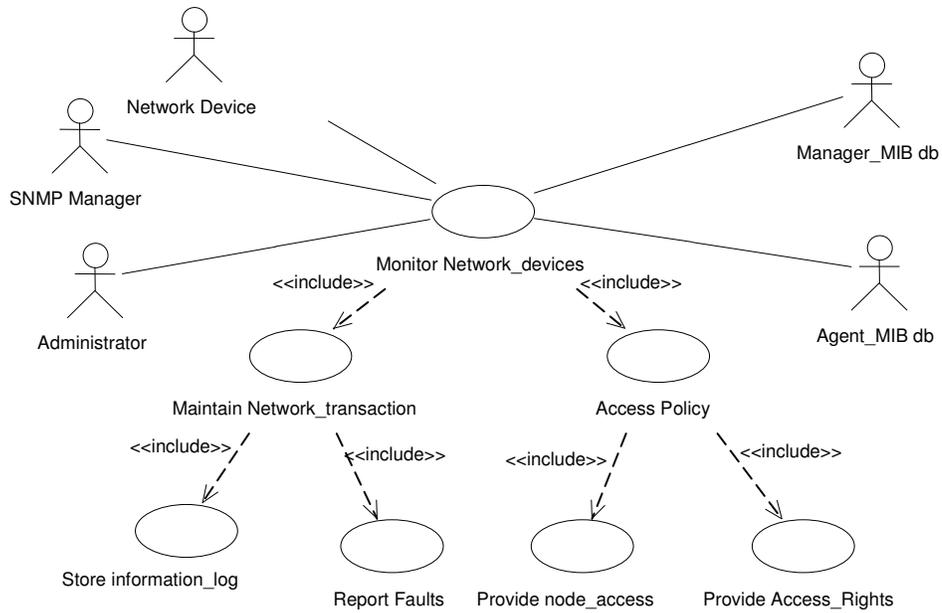


Figure 5.3 RCD Monitor Network\_Devices

<b>USE CASE #</b>	Monitor network devices	
<b>Purpose</b>	Network is monitored to maintain network transaction and to manage access policy. Access policy deals with node access on the network and providing access rights	
<b>Preconditions</b>	Network is initialized.	
<b>Post conditions</b>	Faults are reported and logs are maintained	
<b>Primary Actors</b>	1.SNMP manager 2.Administrator 3.Network device	
<b>Trigger</b>	SNMP manger asking for network information.	
<b>DESCRIPTION</b>	<b>Step</b>	<b>Basic Course of Action</b>
	1	SNMP manager issues a request to access node.
	2	Requested is verified by the agent.
	3	Access is granted to the SNMP manager.
	1	SNMP manager issues a request to access network.
	2	SNMP manager specifies community string
	3	Community string and is verified by the agent.
	4	Access is granted to the SNMP manager.
<b>DESCRIPTION</b>	<b>Step</b>	<b>Alternate Course of Action</b>
	1	Check the device sending request.
	2	Send test packet to the device.
	3	Fault occurs.
	4	Report fault.
<b>DESCRIPTION</b>	<b>Step</b>	<b>Error Scenario</b>
	1	No response is given by device to the test packet

	2	Invalid Community String.
	3	Access rights are violated.

<b>USE CASE #</b>	Maintain network transaction	
<b>Goal</b>	Maintain network transaction	
<b>Purpose</b>	This use case will provide information about how network transaction is handled. It has tasks of reporting faults on detection and store information logs.	
<b>Preconditions</b>	Fault is detected.	
<b>Post conditions</b>	Delivery of status though report to manager	
<b>Primary Actors</b>	1.SNMP manager 2.Administrator 3.Network device	
<b>Trigger</b>	Network transaction occurs.	
<b>DESCRIPTION</b>	<b>Step</b>	<b>Basic Course of Action</b>
	1	Check the device sending request.
	2	Send test packet to the device.
	3	Process the packet returned by the device.
	4	Return status of operation.
	5	Logs of information are stored.
<b>DESCRIPTION</b>	<b>Step</b>	<b>Alternate Course of Action</b>
	1	Check the device sending request.
	2	Send test packet to the device.
	3	Fault occurs.
	4	Report fault.
<b>DESCRIPTION</b>	<b>Step</b>	<b>Error Scenario</b>
	1	No response is given by device to the test packet.

	Managing access rights	
<b>USE CASE #</b>		
Goal	Managing access rights	
Purpose	This use case will provide and manage access rights that will be given to the user of the network.	
Preconditions	User is requesting for access of network information.	
Success Condition	User is given access to the requested information.	
Failed Condition	Access to requested information is denied.	
Post conditions	Privilege levels are defined.	
Primary Actors	1.SNMP manager 2.Administrator 3.Network device	
Trigger	SNMP manger asking for network information.	
<b>DESCRIPTION</b>	<b>Step</b>	<b>Basic Course of Action</b>
	1	SNMP manager issues a request to access network.
	2	Node access request is verified by the agent.
	3	Access is granted to the SNMP manager.
	1	SNMP manager issues a request to access network.
	2	SNMP Manager specifies the community string.
	3	Community string is verified by the agent.
	4	Access is granted to the SNMP manager.
<b>DESCRIPTION</b>	<b>Step</b>	<b>Error Scenario</b>
	1	Request time-out.
	2	Access rights are violated.
	3	Community string is invalid

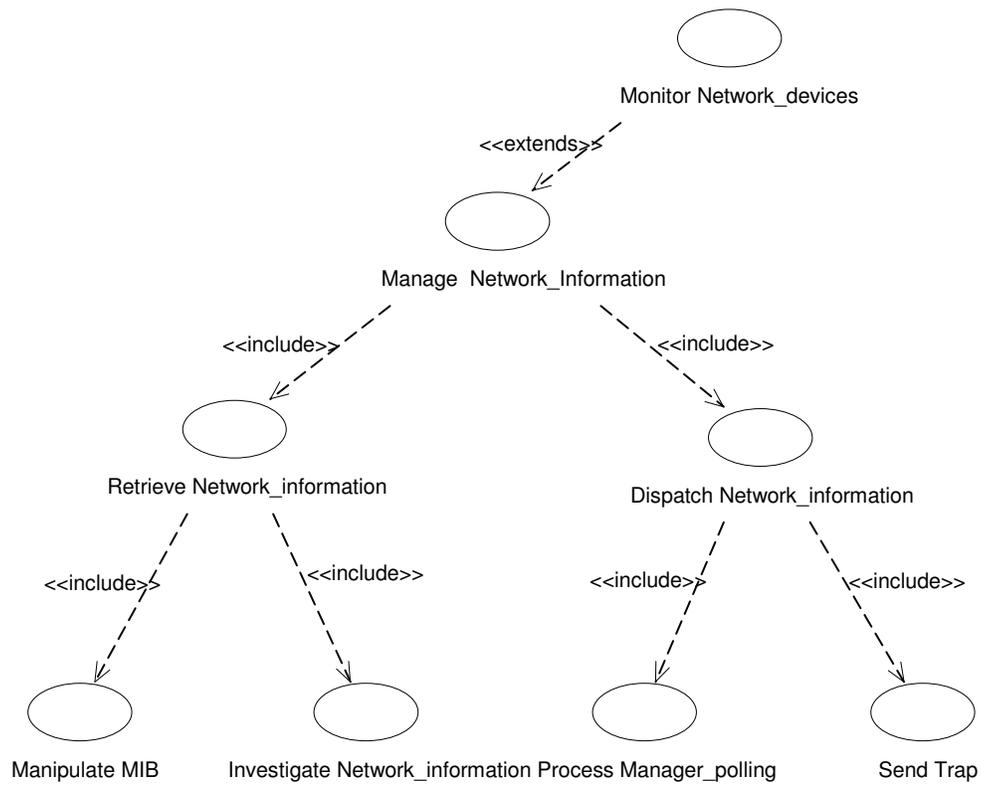


Figure 5.4 RCD Manage Network\_Infomration

<b>USE CASE #</b>	Manage Network Information	
<b>Purpose</b>	Retrieving and dispatching information on request of SNMP Manager is vital task in network management. SNMP manager requests SNMP agent for information, agent retrieves it from MIBs. Agent continuously traces network and sends trap to manager if fault is found.	
<b>Preconditions</b>	User is requesting for access of network information.	
<b>Post conditions</b>	MIBS are manipulated as per requirement	
<b>Primary Actors</b>	1.SNMP manager 2.Administrator 3.Network device	
<b>Trigger</b>	SNMP Manager requests SNMP agent	
<b>DESCRIPTION</b>	<b>Step</b>	<b>Basic Course of Action</b>
	1	SNMP manager send request for dispatching information.
	2	Request is processed by the agent.
	3	Data is verified and MIBs are manipulated.
	4	Status is sent to the SNMP manager by the agent.
	1	SNMP manager sends request for network information.
	2	Request sent is processed by the agent.
	3	Information is searched in the MIB.
	4	Corresponding OIDs are returned and response PDU is created.
	5	Response PDU is sent by the agent.
<b>DESCRIPTION</b>	<b>Step</b>	<b>Alternate Course of Action</b>
	1	Fault or change in OID value is noticed by SNMP agent.
	2	Trap is sent to SNMP manager.
	3	SNMP Manager takes the required actions.
	1	SNMP Agent queries MIBs for OID values after regular intervals of time.
	2	If there are any changes ,SNMP Agent creates the Response PDU and sends the changed OID values
	3	Response PDU along with trap is sent by the agent to the manager.
<b>DESCRIPTION</b>	<b>Step</b>	<b>Error Scenario</b>
	1	Invalid OID.

	2	Write permission Denied.
	3	Request timed out.
	4	Object instance not found

<b>USE CASE #</b>	Dispatch network information.	
<b>Goal</b>	Dispatch network information.	
<b>Purpose</b>	This use case will dispatch network information to the SNMP manager.	
<b>Preconditions</b>	Manager requests for network information or fault occurs in network operations	
<b>Post conditions</b>	Requested information is delivered to SNMP manager	
<b>Primary Actors</b>	1.SNMP manager 2.Administrator 3.Network device	
<b>Trigger</b>	SNMP Manager requests SNMP agent	
<b>DESCRIPTION</b>	<b>Step</b>	<b>Basic Course of Action</b>
	1	SNMP manager send request for dispatching information.
	2	Request is processed by the agent.
	3	Data is verified and MIBs are manipulated.
	4	Status is sent to the SNMP manager by the agent.
<b>DESCRIPTION</b>	<b>Step</b>	<b>Alternate Course of Action</b>
	1	Fault or change in OID value is noticed by SNMP agent.
	2	Trap is sent to SNMP manager.
	3	SNMP Manager takes the required actions.
<b>DESCRIPTION</b>	<b>Step</b>	<b>Error Scenario</b>
	1	Write permission denied.
	2	Invalid OID.

<b>USE CASE #</b>	Retrieve Information	
<b>Goal</b>	Retrieve Information	
<b>Purpose</b>	This use case will elaborate on how information retrieval is done.	
<b>Preconditions</b>	Manager dispatches network information for particular objects	
<b>Post conditions</b>	Required objects are modified in the MIB.	
<b>Primary Actors</b>	1.SNMP manager 2.Administrator 3.Network device	
<b>Trigger</b>	Request from SNMP manager for information retrieval.	
<b>DESCRIPTION</b>	<b>Step</b>	<b>Basic Course of Action</b>
	1	SNMP manager sends request for network information.
	2	Request sent is processed by the agent.
	3	Information is searched in the MIB.
	4	Corresponding OIDs are returned and response PDU is created.
	5	Response PDU is sent by the agent.
<b>DESCRIPTION</b>	<b>Step</b>	<b>Alternate Course of Action</b>
	1	SNMP Agent queries MIBs for OID values after regular intervals of time.
	2	If there are any changes ,SNMP Agent creates the Response PDU and sends the changed OID values
	3	Response PDU along with trap is sent by the agent to the manager.
<b>DESCRIPTION</b>	<b>Step</b>	<b>Error Scenario</b>
	1	Request timed-out.
	2	Object Instance not Found.
	3	Invalid OID.

**SYMANTEC**

Project Title: LEAF-SNMP

## SEQUENCE DIAGRAMS

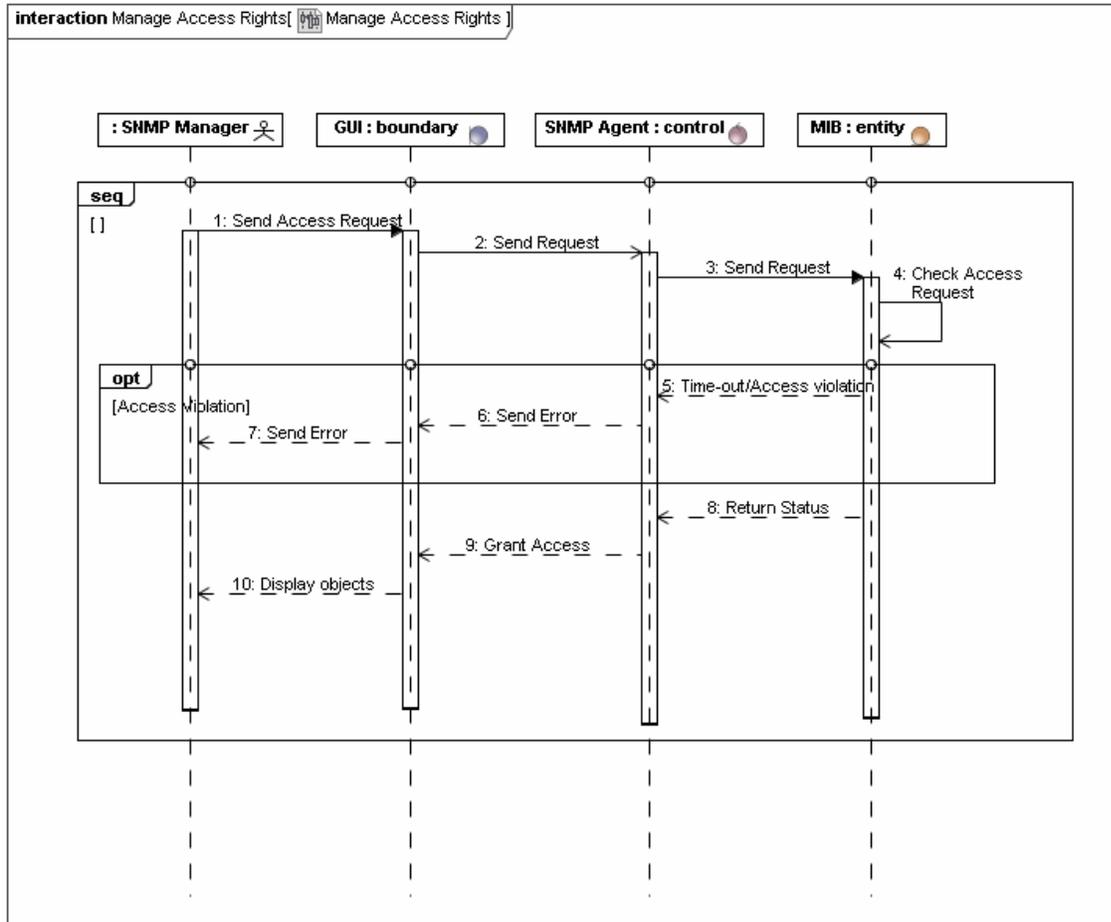
## SEQUENCE DIAGRAM OVERVIEW

### SCENARIO DESCRIPTION

Informational Item	Information
Use Case	Manage Access rights
Scenario Name	SNMP Agent requesting to access node.
Steps	1) SNMP manager issues a request or sends community string to access network. 2) Node Access or community string is verified by the agent. 3) Access is granted to the SNMP manager.

### MESSAGE DESCRIPTION

Message	Type	From Object	To Object
Send Access Request	Request	SNMP Manager	SNMP Agent
Check Access Request	Self	MIB	To self
Return Status	Response	MIB	SNMP Agent
Grant Access	Response	SNMP Agent	SNMP Manager



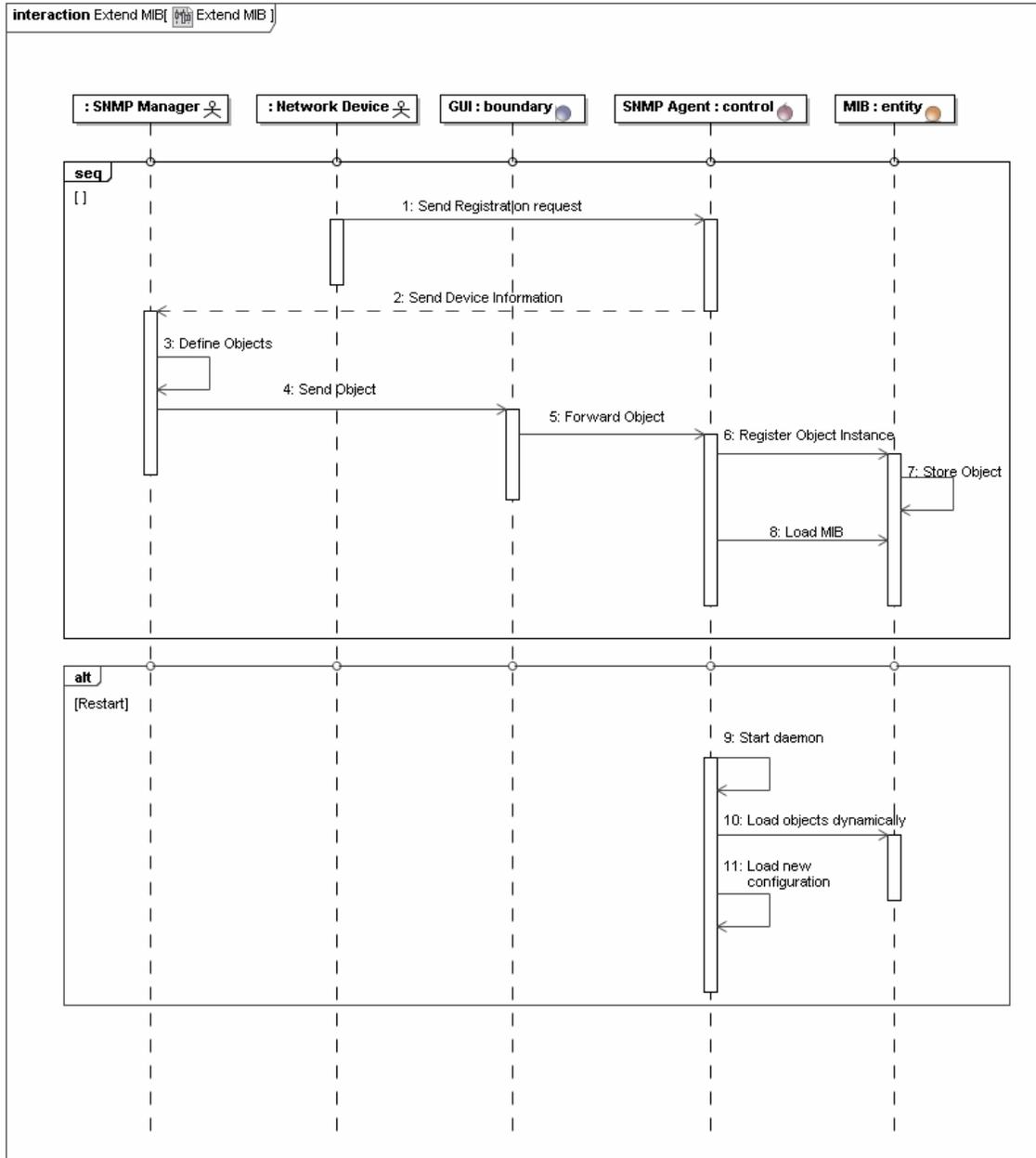
**Figure 6.1 Behaviour:: Manage Access Rights**

## SCENARIO DESCRIPTION

Informational Item	Information
Use Case	Extend Management_Info_Base
Scenario Name	Extending a MIB
Steps	1) Device sends request for registration 2) Define objects in the MIB according to ASN. 3) Store the MIB file in the mibs directory. 4) It completes the registration procedure. 5) MIB is loaded.

## MESSAGE DESCRIPTION

Message	Type	From Object	To Object
Send registration request	Request	Network device	SNMP Agent
Send Device Info	Response	SNMP agent	SNMP Manager
Define Object	Self	SNMP Manager	To self
Send Object	Request	SNMP Manager	SNMP Agent
Register Instance	Request	SNMP Agent	MIB
Load MIB	Request	SNMP Agent	MIB



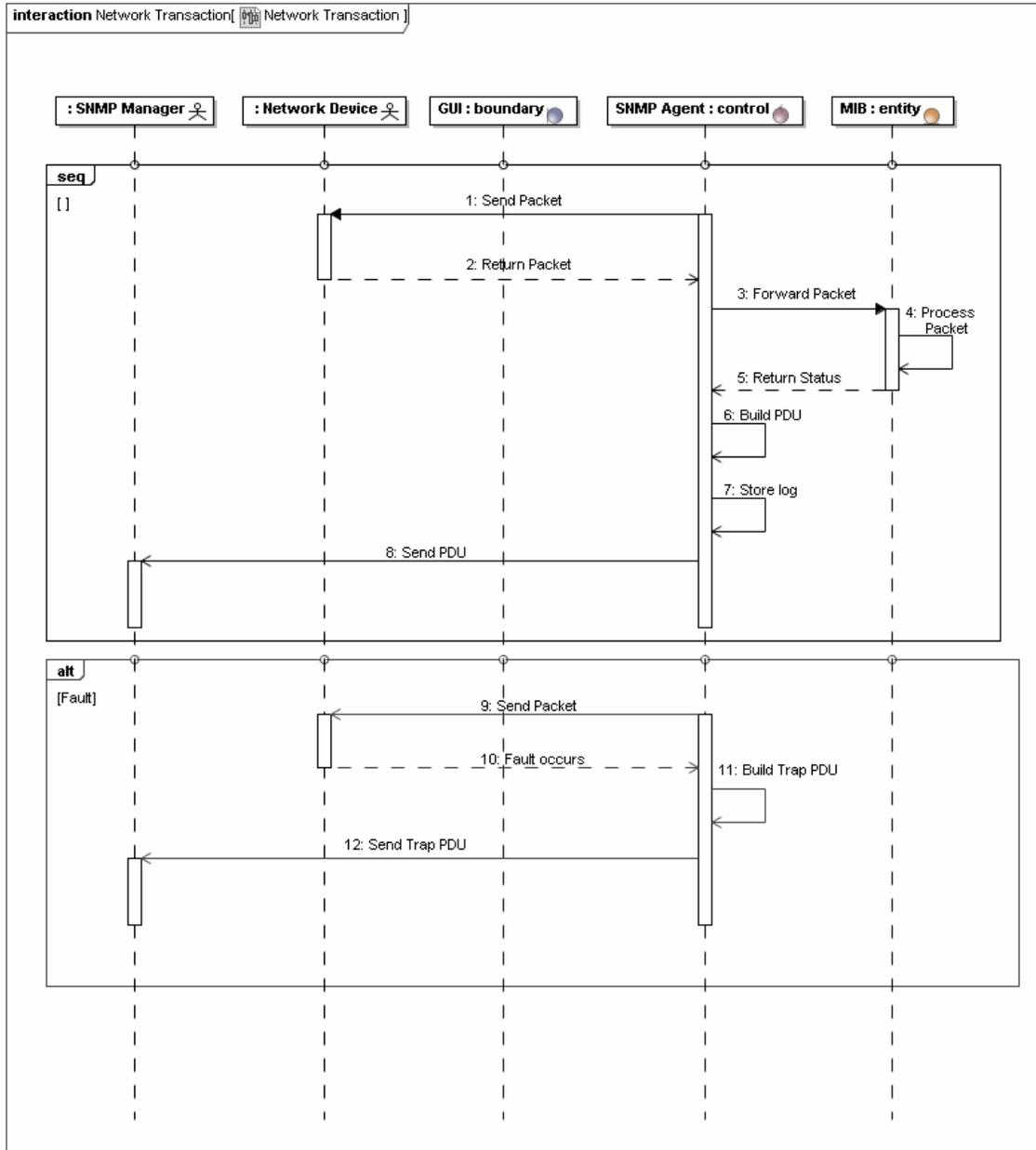
**Figure 6.2 Behaviour:: Extend MIBs**

## SCENARIO DESCRIPTION

Informational Item	Information
Use Case	Maintain network transaction
Scenario Name	Device testing.
Steps	1) SNMP agent will check the device. 2) Then it will send test packet to the device. 3) SNMP agent processes packet returned from Device 4) Log of information is stored by the agent. 5) SNMP Agent sends trap to the manager.

## MESSAGE DESCRIPTION

Message	Type	From Object	To Object
Send test packet	Request	SNMP agent	Network Device
Return packet	Response	Network Device	SNMP agent
Process packet	Self	SNMP agent	To self
Build trap PDU	Self	SNMP Agent	To self
Store Logs	Self	SNMP Agent	To self
Send trap	Response	SNMP Agent	SNMP manager



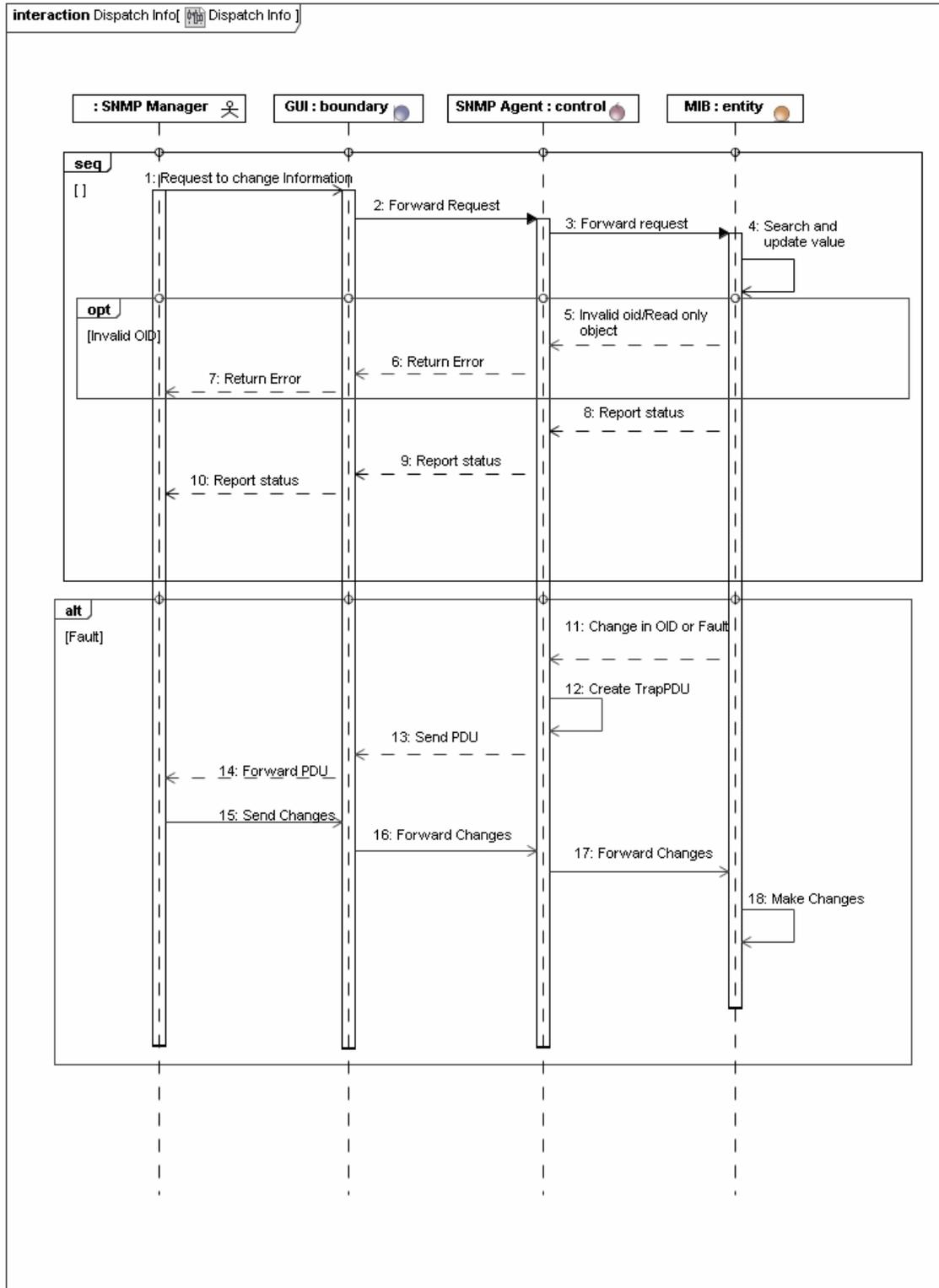
**Figure 6.3 Behaviour: Maintain Network Transition**

## SCENARIO DESCRIPTION

Informational Item	Information
Use Case	Dispatch information
Scenario Name	SNMP agent dispatching information to the devices.
Steps	<ol style="list-style-type: none"><li>1) SNMP Manager sends request to modify particular data.</li><li>2) The request is processed by the agent.</li><li>3) SNMP agent queries the MIBs with the particular OIDs.</li><li>4) The OID values are changed in the corresponding MIBs.</li><li>5) Corresponding Status is sent by the agent.</li></ol>

## MESSAGE DESCRIPTION

Message	Type	From Object	To Object
Send Request	Request	SNMP manager	SNMP Agent
Querying the MIBs	Request	SNMP Agent	MIB
Return status	Reply	SNMP Agent	SNMP manager
Change MIBs	Reply	MIB	To self



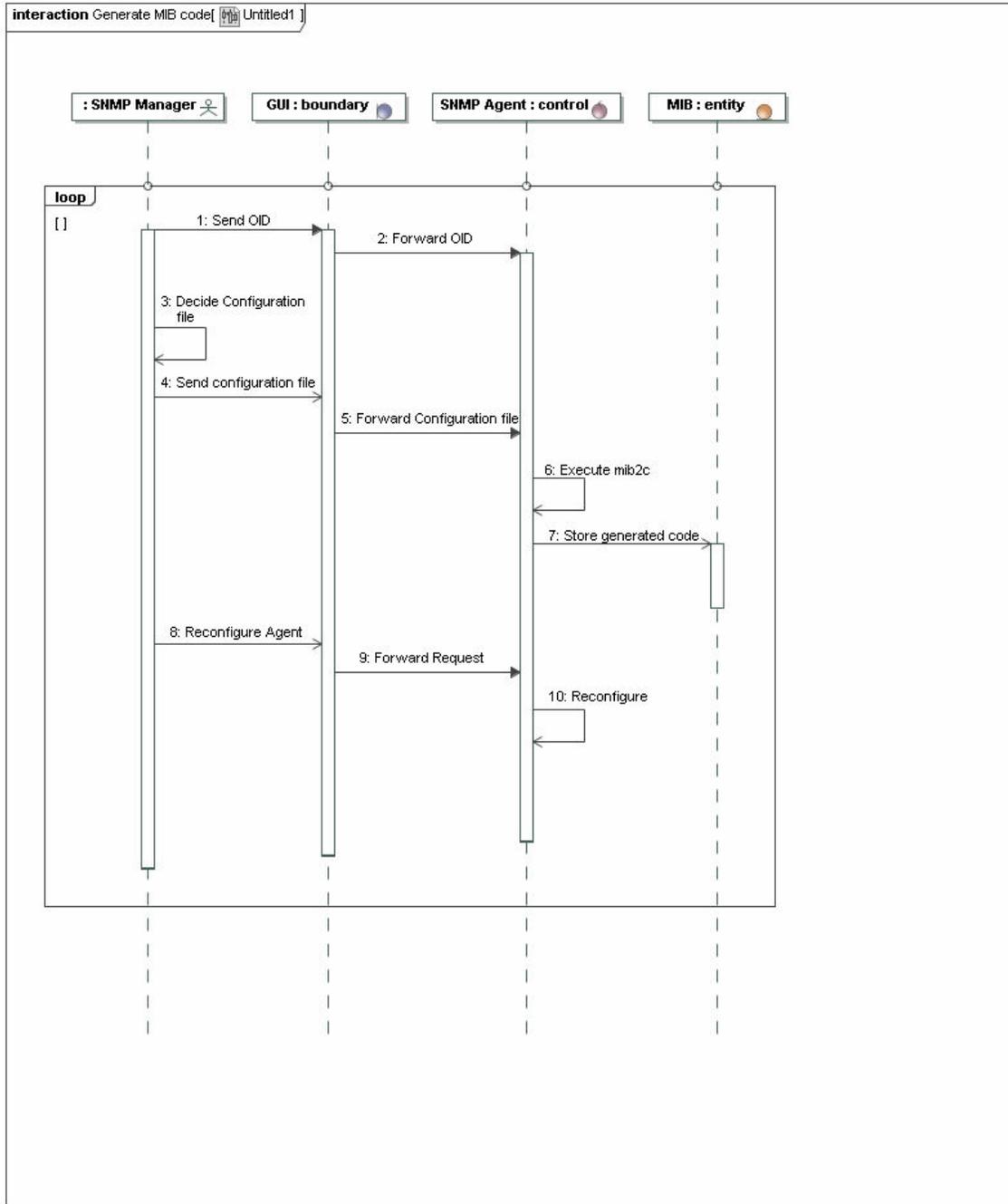
**Figure 6.4 Behaviour: Dispatch Information**

## SCENARIO DESCRIPTION

Informational Item	Information
Use Case	Generate MIB_Code
Scenario Name	The MIB code and header files are generated and MIB is embedded in agent.
Steps	1) SNMP Manager sends proper OID to SNMP Agent. 2) SNMP Manager decides and sends configuration file to Agent. 3) SNMP Agent executes mib2c 4) The generated code is saved in the MIBs directory. 5) SNMP Manager reconfigures SNMP Agent with the newly added MIB.

## MESSAGE DESCRIPTION

Message	Type	From Object	To Object
Send proper OID	Request	SNMP Manager	SNMP Agent
Decide Configuration File	Self	SNMP Manager	To self
Send Configuration File	Request	SNMP Manager	SNMP Agent
Execute mib2c	Self	SNMP agent	To Self
Store Generated Code	Request	SNMP Agent	MIB
Reconfigure agent	Request	SNMP Manager	SNMP Agent



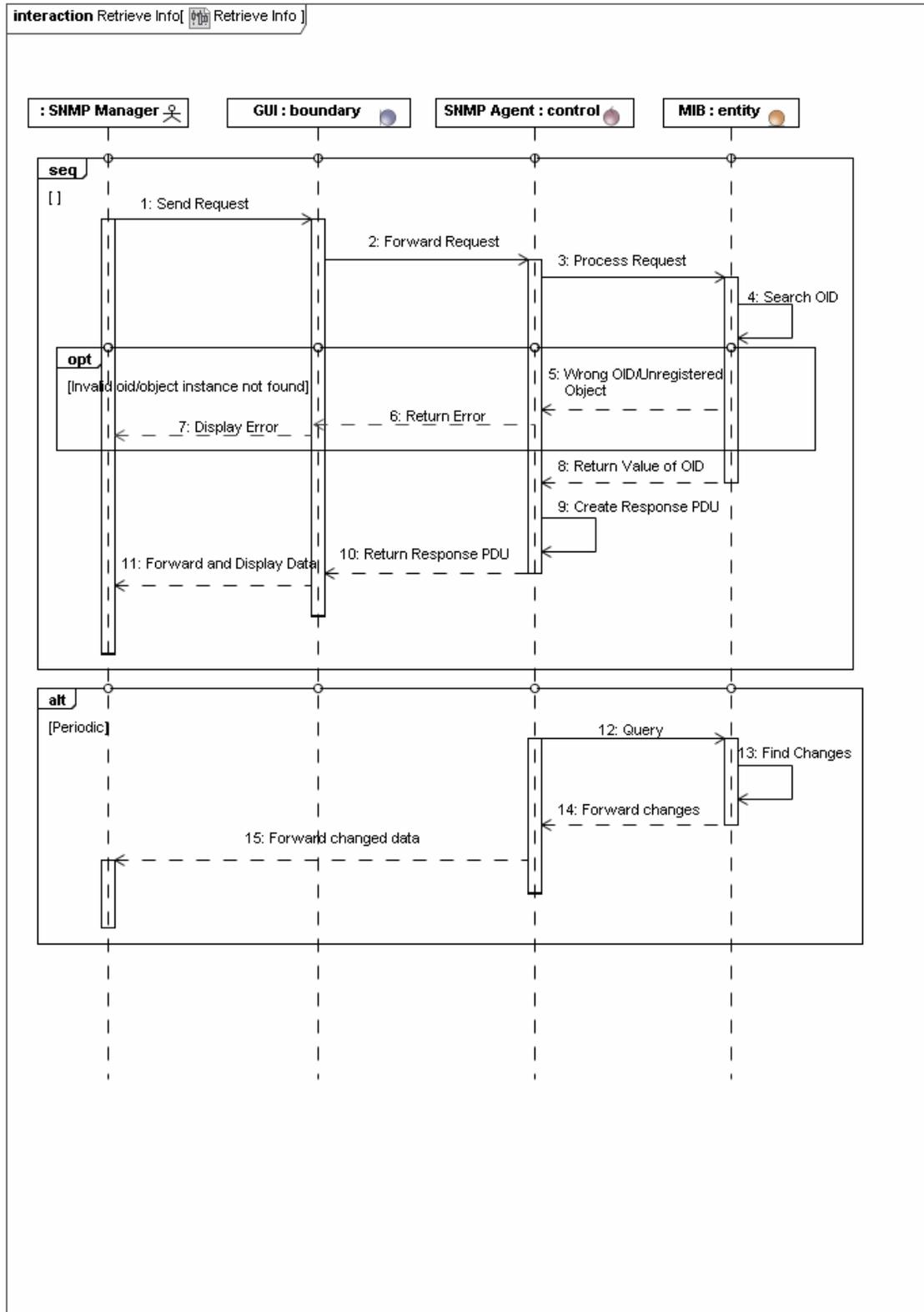
## 5.5 Behaviour :: Generate MIB Code

## SCENARIO DESCRIPTION

Informational Item	Information
Use Case	Retrieve Information.
Scenario Name	SNMP agent retrieving information from MIB.
Steps	<ol style="list-style-type: none"><li>1) SNMP manager sends request for information.</li><li>2) The request sent by manager is processed by the agent.</li><li>3) The information is searched in the MIB by the agent.</li><li>4) Corresponding OIDs are returned and response PDU is built.</li><li>5) Success or failure is reported by the agent.</li></ol>

## MESSAGE DESCRIPTION

Message	Type	From Object	To Object
Send request	Request	SNMP Manager	SNMP Agent
Return OID status	Response	MIB	SNMP Agent
Investigate information	Self	SNMP Agent	MIB
Return of OIDs	Response	MIB	SNMP Agent
Return status	Response	SNMP Agent	SNMP Manager



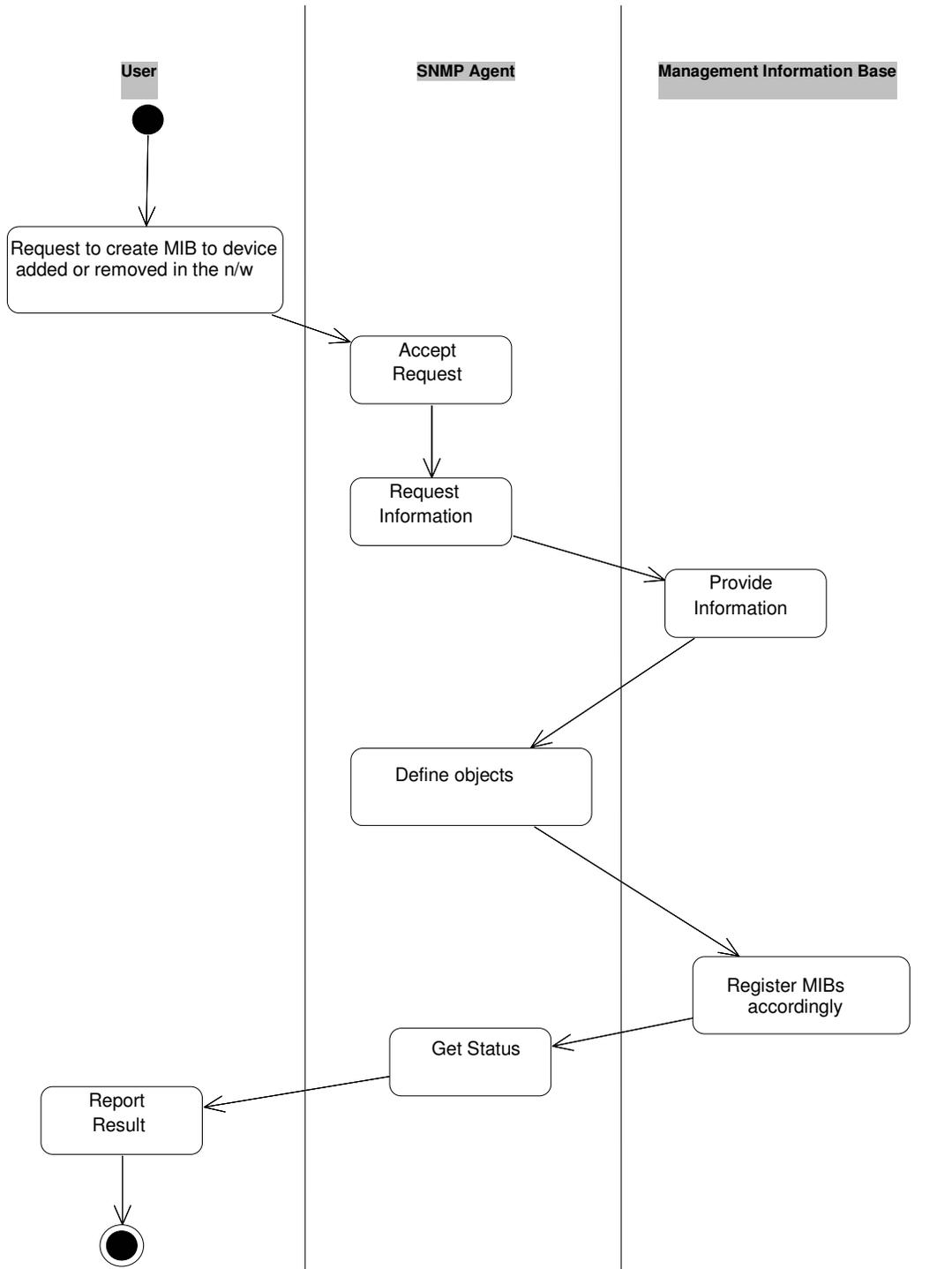
**Figure 5.6 Behaviour :: Retrieve Network Information**

**SYMANTEC**

Project Title : LEAF-SNMP

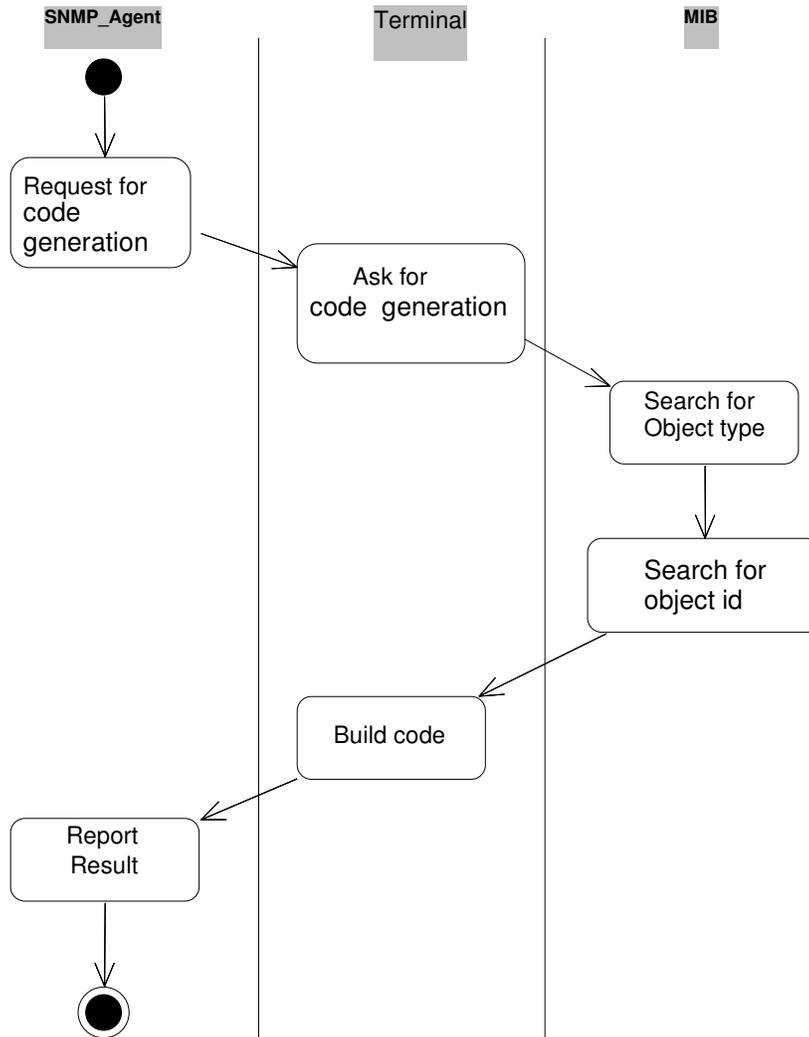
## ACTIVITY DIAGRAMS

# 1) EXTEND MANAGEMENT\_INFO\_BASE



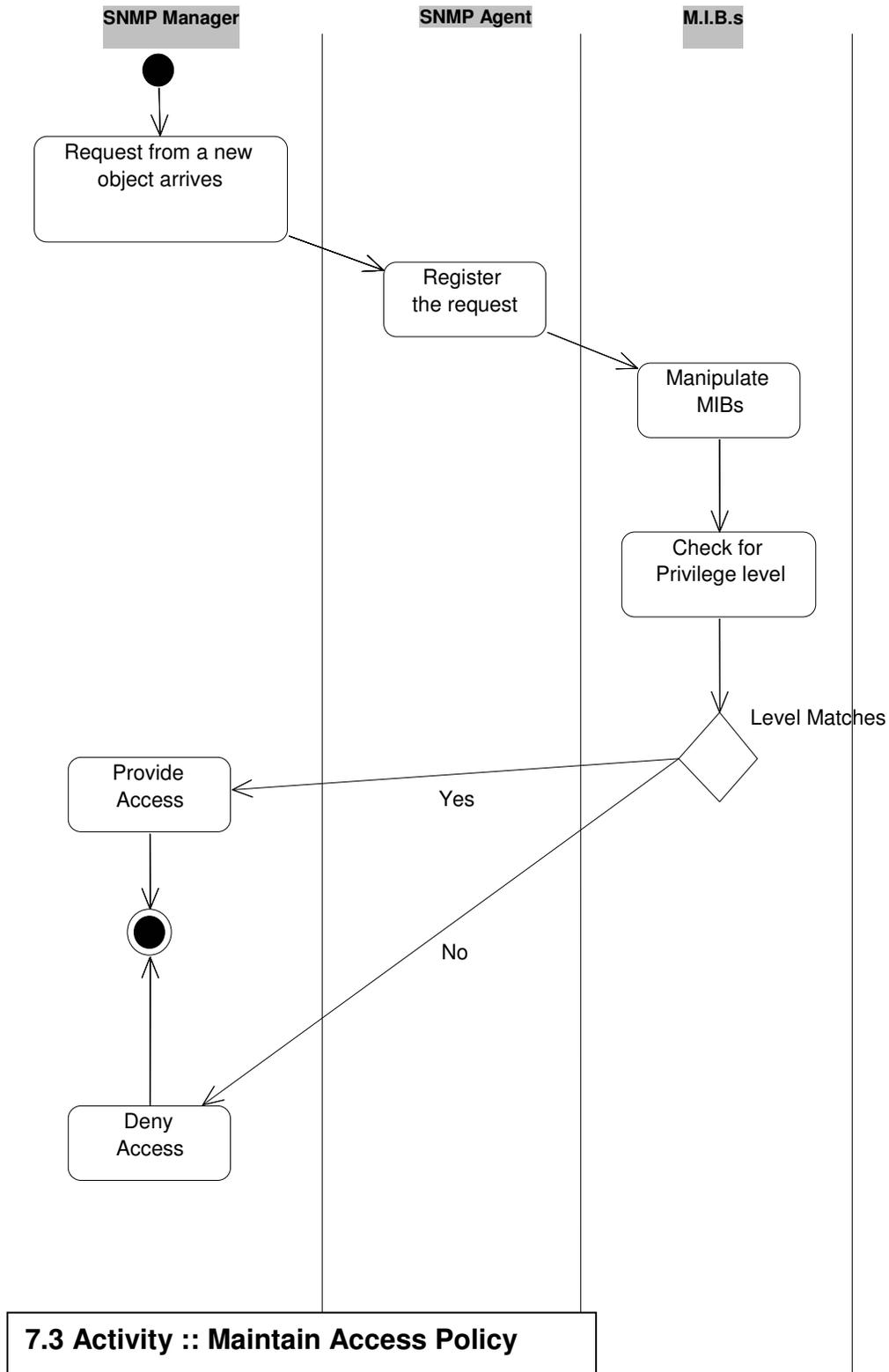
**7.1 Activity :: Extend Management\_Information\_Base**

## 2) Generate MIB code

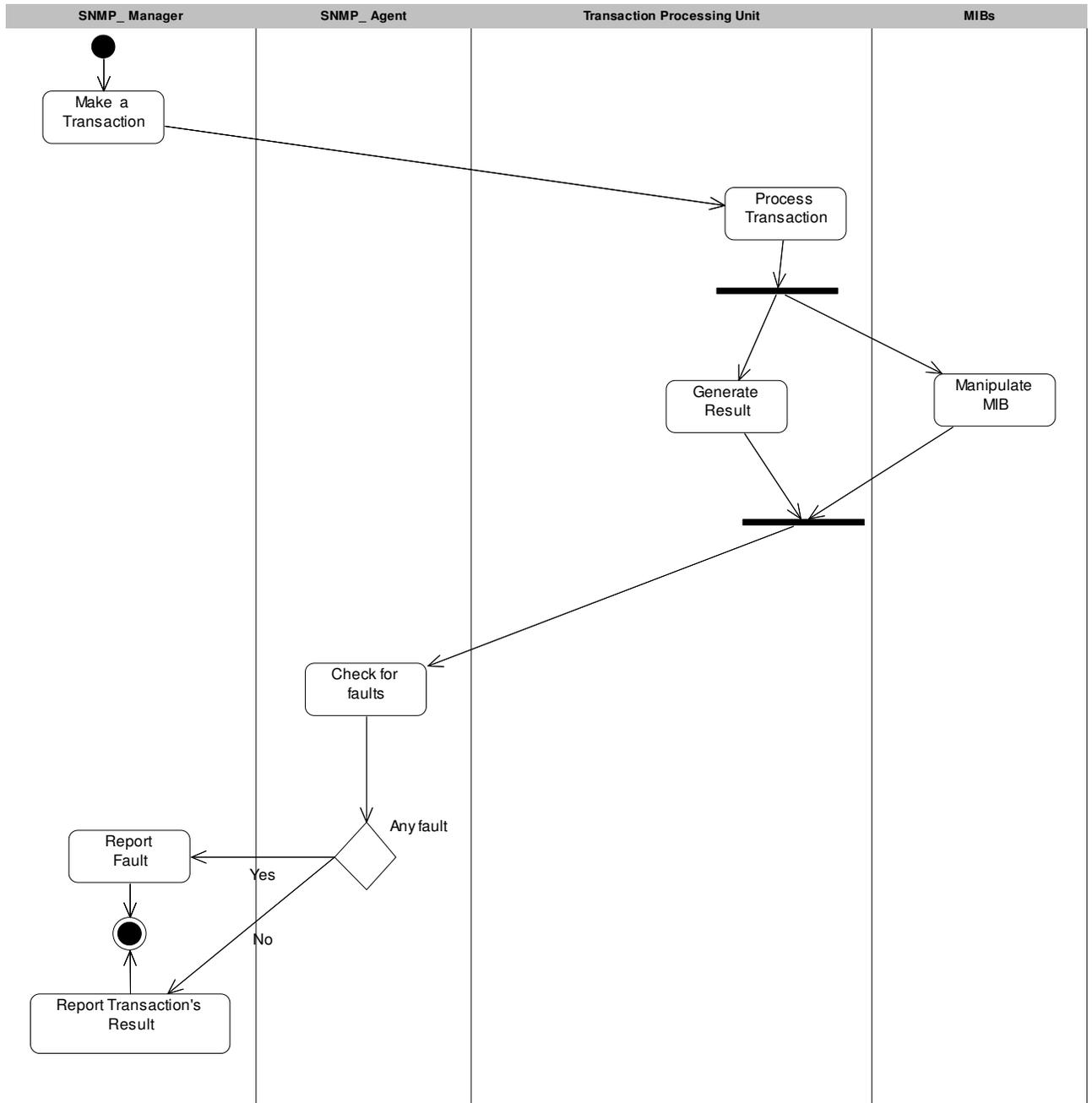


**7.2 Activity :: Generate MIB\_Code**

### 3) Maintain Access Policy

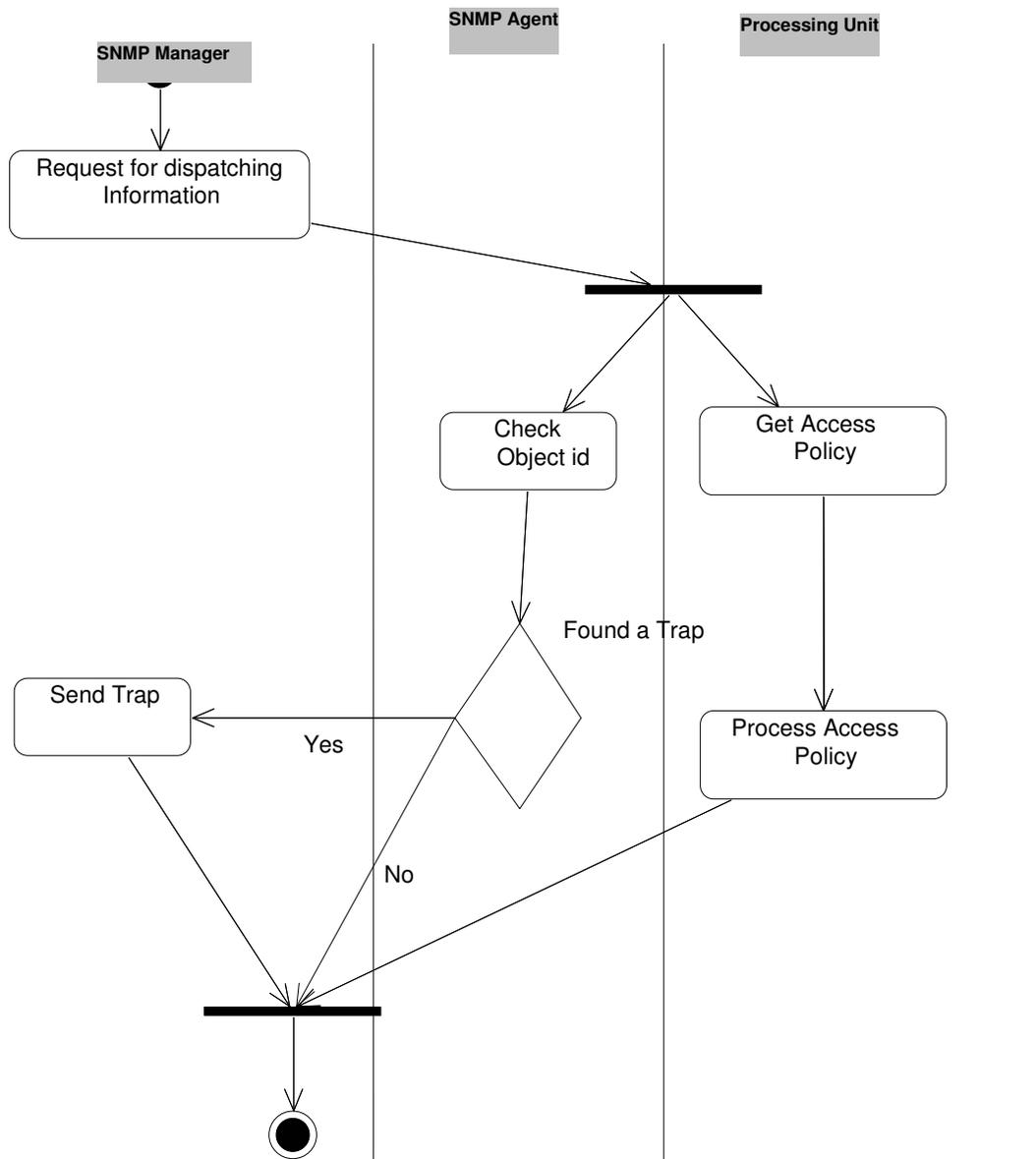


## 4) Maintain Network Transaction



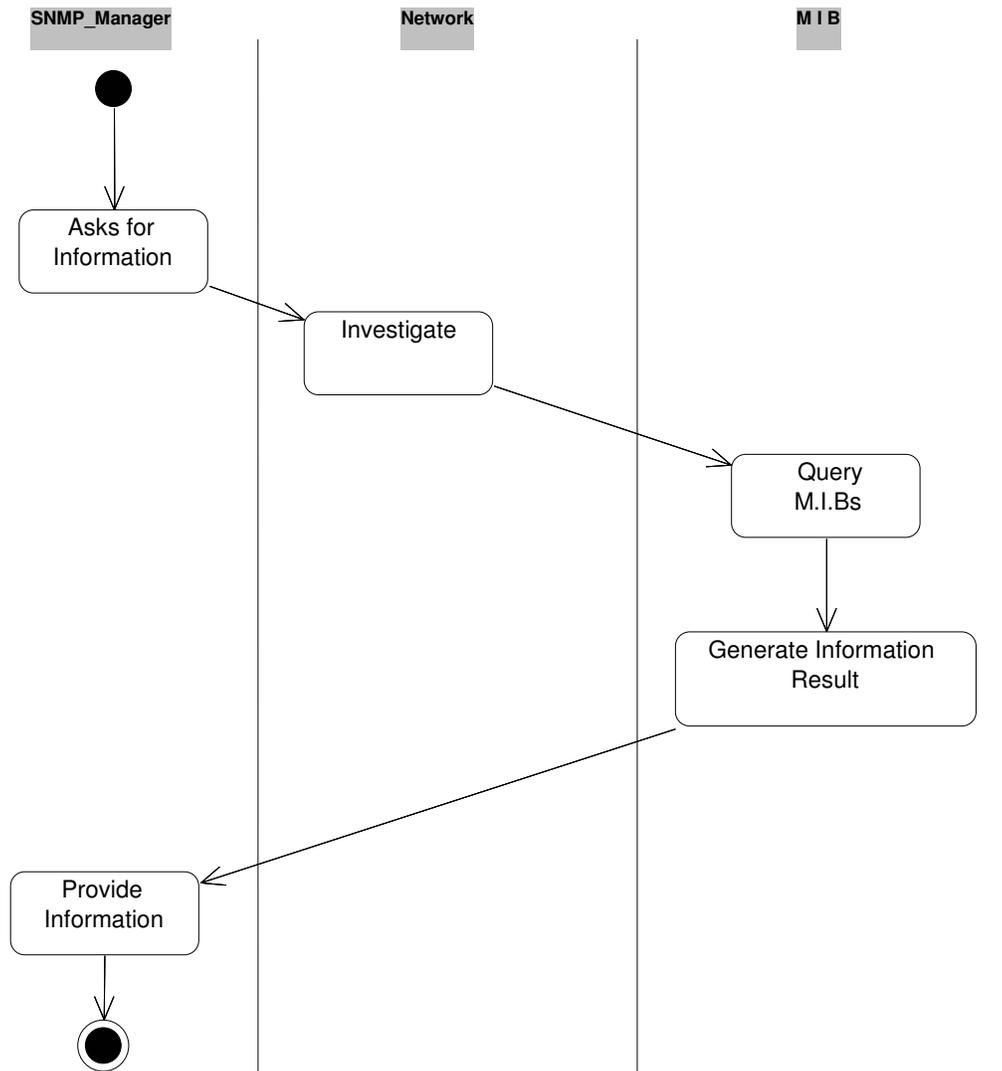
### 7.4 Activity :: Maintain Network Transaction

## 5) Dispatch Network Information



### 7.5 Dispatch Network Information

## 6) Retrieve Network Information



### 7.6 Retrieve Network Information

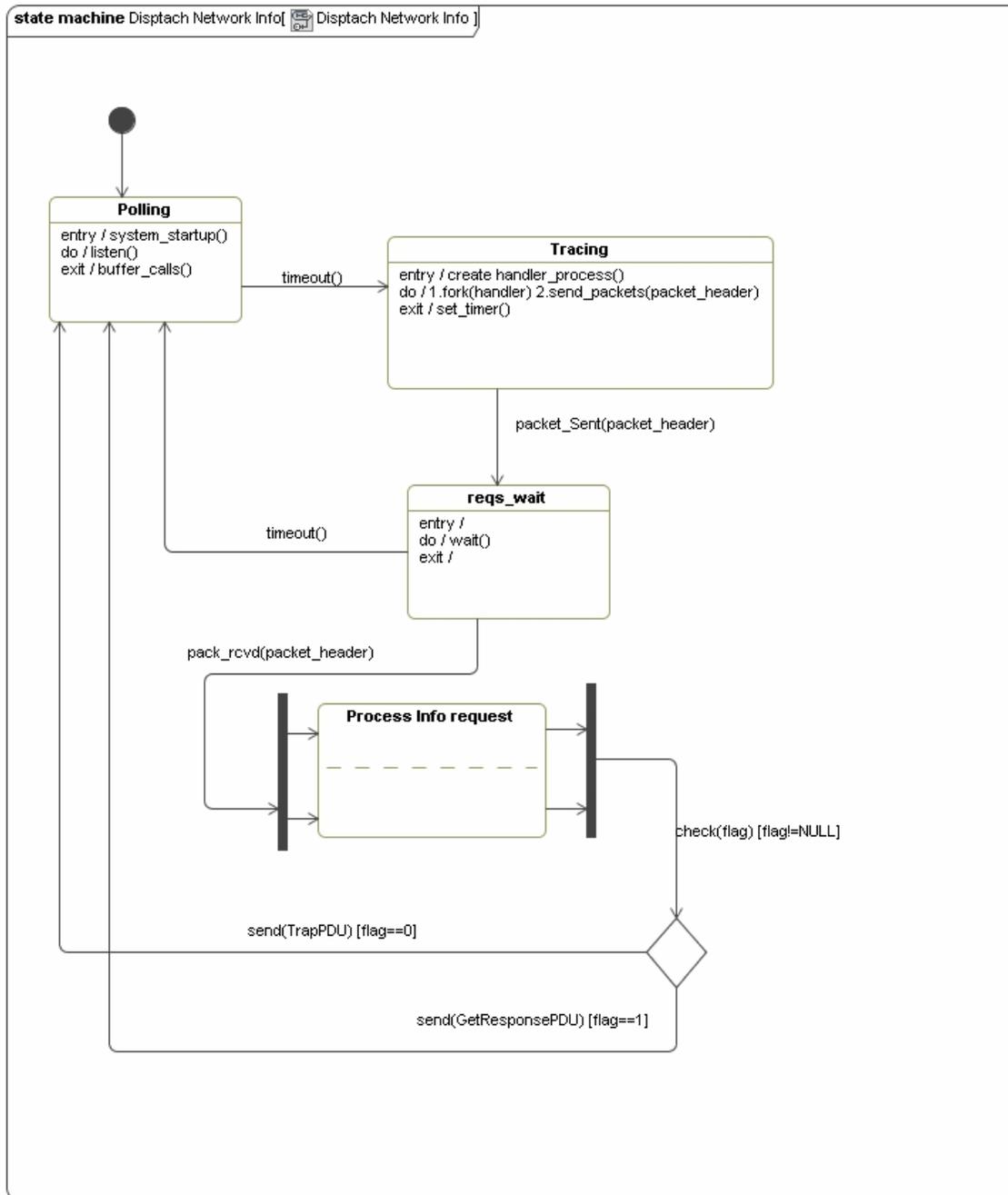
**SYMANTEC**

Project Title: LEAF-SNMP

## STATE CHART DIAGRAMS

## STATE CHART DIAGRAM OVERVIEW

### 1) Dispatch Network Information



**Figure 8.1 State Chart :: Dispatch Network Information**

## 2) Manage Access Rights

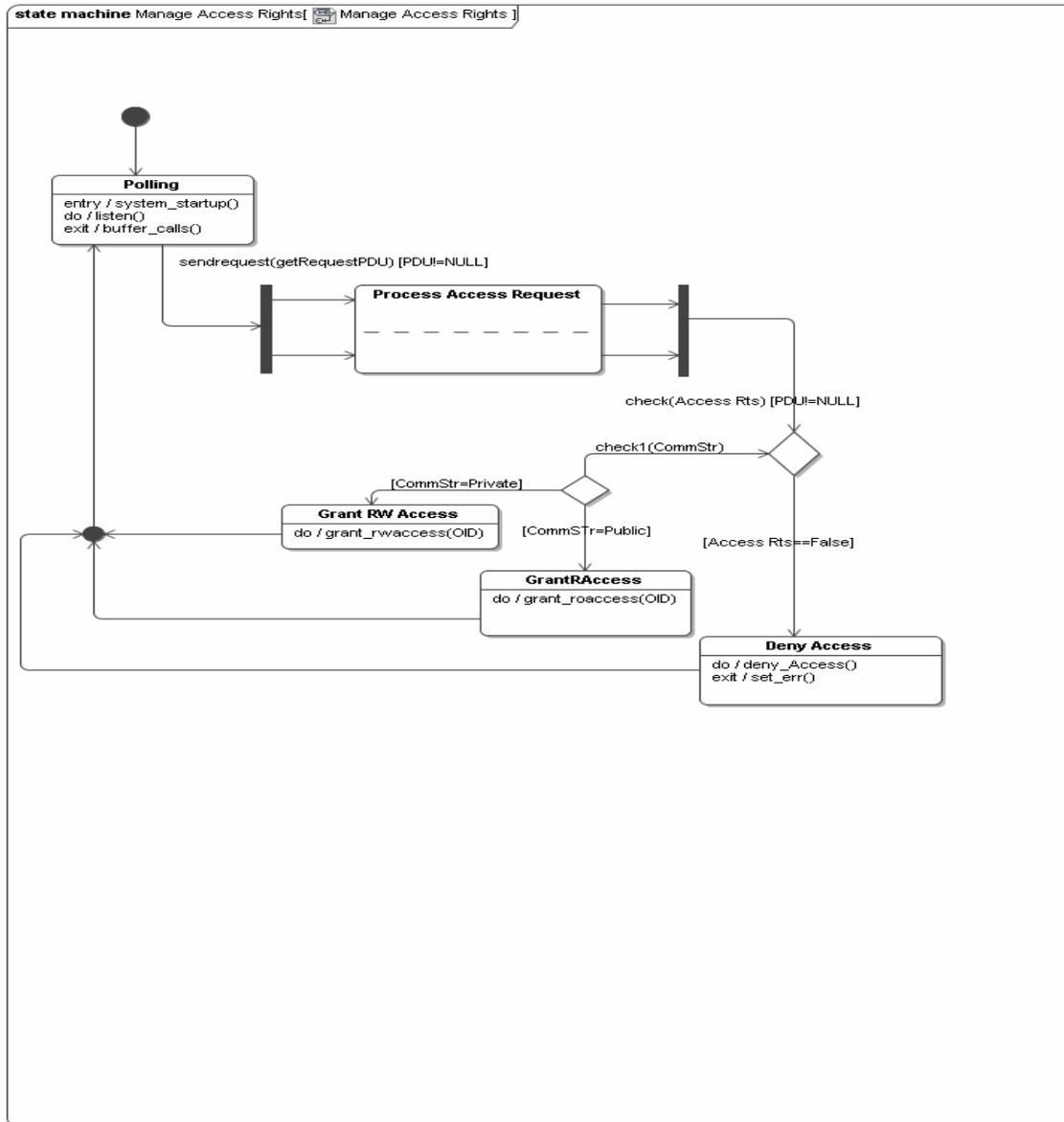
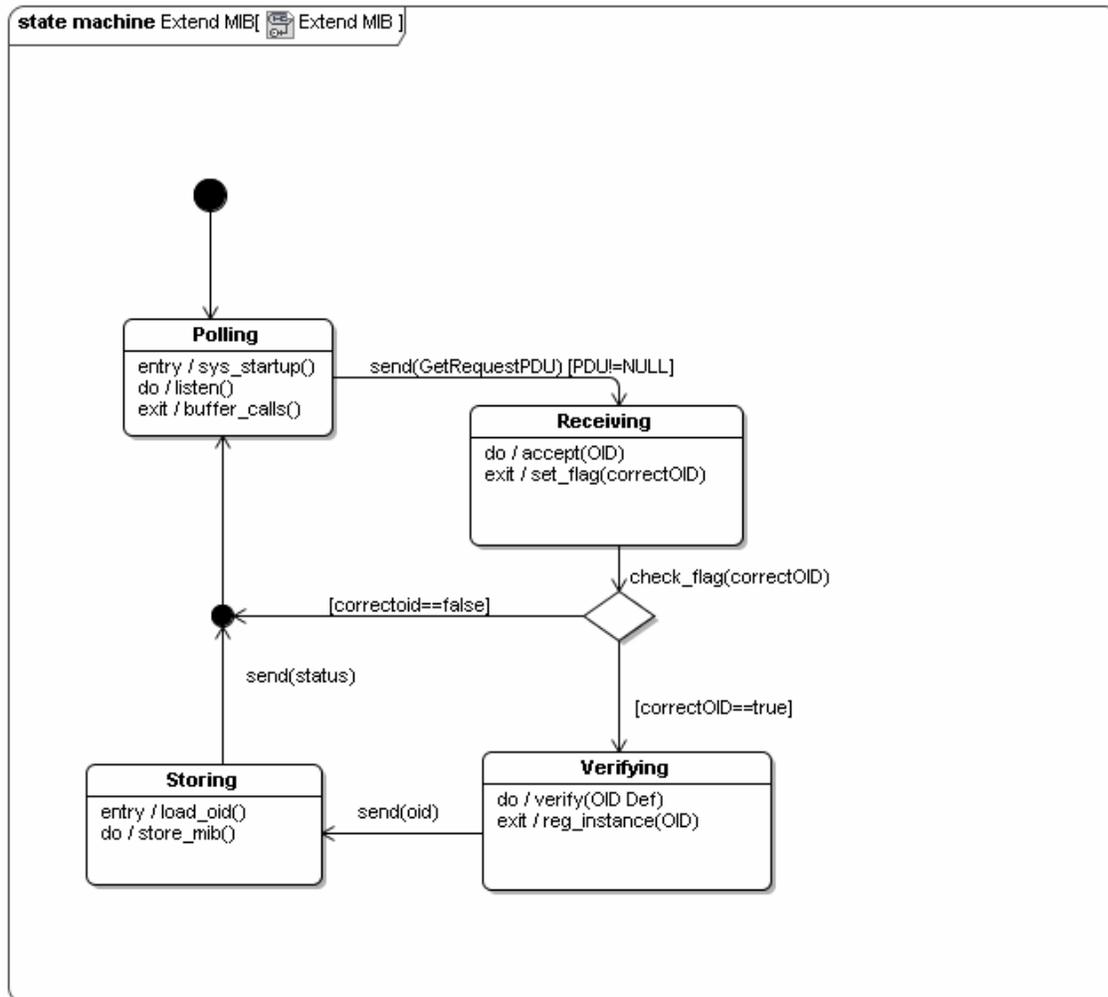


Figure 8.2 State Chart :: Manage Access Rights

### 3) Extend MIB



**Figure 8.3 State Chart :: Extend MIB**

#### 4) Generate MIB\_Code

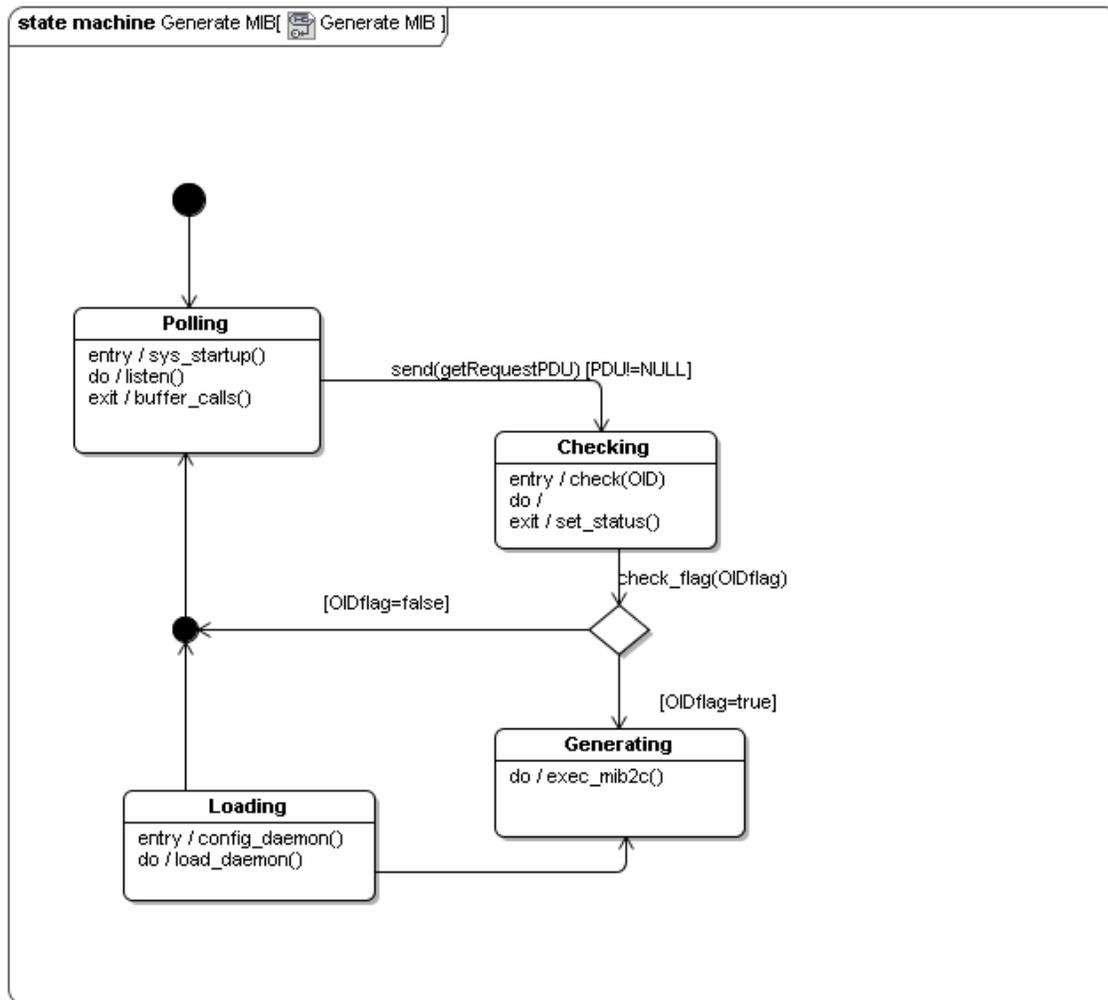
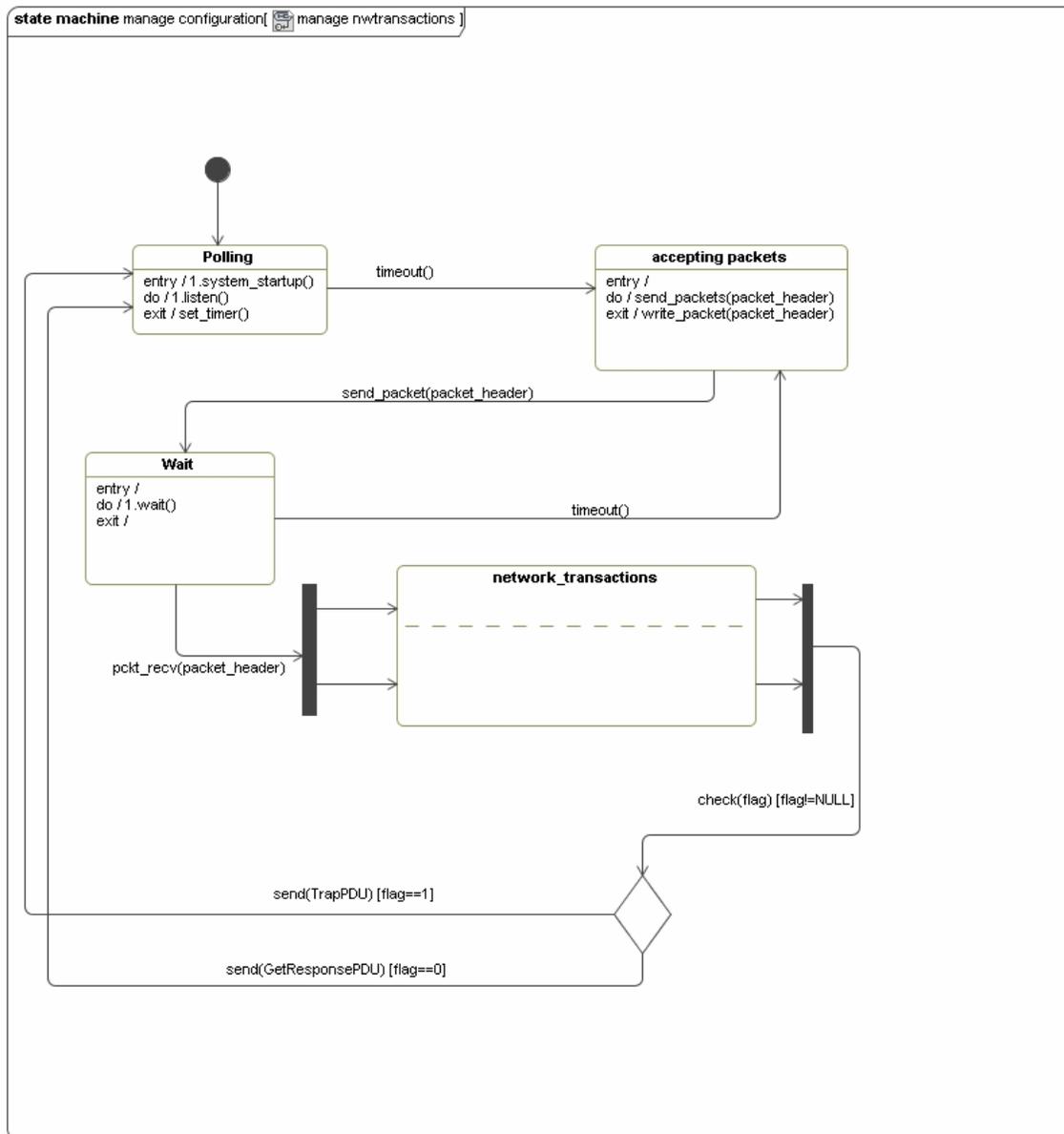


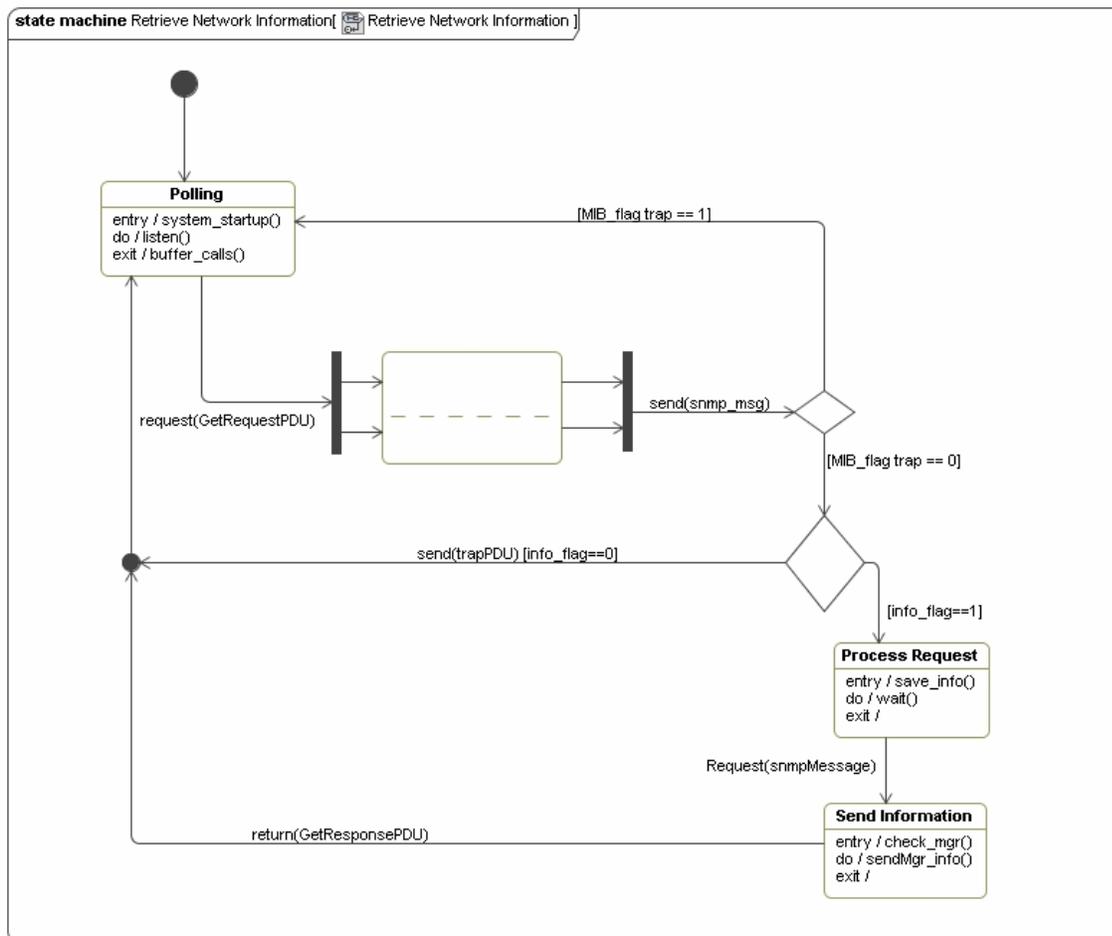
Figure 8.4 State Chart :: Generate MIB Code

## 5) Maintain Network Transaction



**Figure 8.5 State Chart :: Maintain Network Transaction**

## 6) Retrieve Network Information



**Figure 8.6 State Chart :: Retrieve Network Information**

**SYMANTEC**

**Project Title: LEAF-SNMP**

# **SYSTEM DESIGN DOCUMENT**

## CLASS DIAGRAM DESCRIPTION

### CRC TEMPLATE

Class Name	Network
Super class	None
Subclass	Network device
VARIABLES	network Type, network Topology, networkId.
Services	getData(), setData().

Class Name	NetworkDevice
Superclass	Network
VARIABLES	deviceId, address, adjacentNodes

Class Name	SNMPAgent
Superclass	EMS
Subclass	NetworkInformation, TraceLink, MonitorDevices
VARIABLES	VersionNumber
Services	sendTrap(), sendData().

Class Name	GUI
Subclass	SNMPAgent
Services	receiveData(), displayData().

Class Name	NetworkInformaton
VARIABLES	fault No., faultInfo
Services	reportFault(), replyPolling(), manageConfig(), accountingDevice().

Class Name	ExtendMib
Superclass	SNMPAgent
VARIABLES	MIB
Services	generateCode

Class Name	MonitorDevices
Superclass	SNMPAgent
VARIABLES	nodeList, username, password.
Services	searchError()
Responsibilities	Collaborators

Class Name	Session
Subclass	SNMPSession
VARIABLES	sessionId
Services	getState(), close(), open().
Responsibilities	Collaborators

Class Name	SNMPSession
VARIABLES	snmpInformation
Services	Send(), Snmpget(), snmpSet().

Class Name	SNMPTrapSession
Services	waitForTrap()
Responsibilities	Collaborators

Class Name	SNMP PDU
Responsibilities	Collaborators

Class Name	SNMPDataType
Services	GetType(), Copy()
Responsibilities	Collaborators

# CLASS DIAGRAM

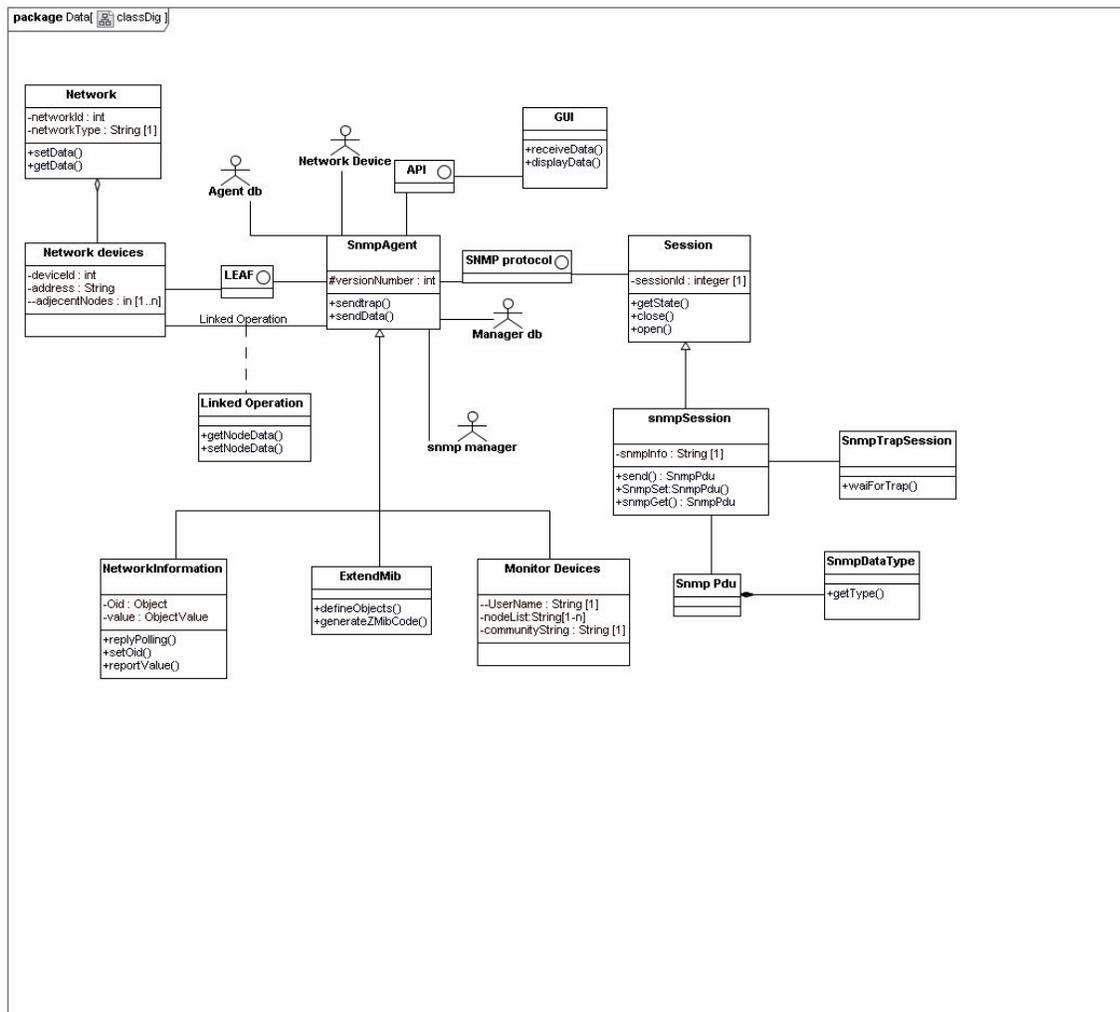


Figure 9.1 Class Diagram

**Symantec**

Project Title: LEAF-SNMP

# SYSTEM IMPLEMENTATION DOCUMENT

---

## COMPONENT DIAGRAM DESCRIPTION

### 1. GUI

<b>Attribute</b>	<b>Description</b>
Classification	Interface
Definition	To provide a graphical user interface to the SNMP Manager
Responsibilities	The component is responsible for providing a GUI which provides various views like manager and user. It provides functionality to retrieve and set various parameters and also display network information.
Constraints	JDK should be installed on the system.
Composition	Manager View : To provide expert options like modifying data, extending MIB in the GUI User View : To provide facility to read the network information
Uses/Interactions	The component uses the SNMP PDUs interface to send requests from the manager to the agent in the form of PDUs. It also uses the lrpmodules interface to read about the network, package and system information.
Resources	AdventNetLogging.jar, Applet Viewer, Browser
Processing	It retrieves the information about the network variables. It also sets the data for particular network variables. <b>Refer to PDL 1</b>
Interface/Exports	Displaydata : It displays the data related to the network variables Setdata : It sets the data related to the particular network variable.

## PDL 1

/\* The following procedure handles the management of the network information when a request is made by the SNMP Manager. During its execution it calls two procedures namely get\_networkinfo() and set\_networkinfo()\*/

### PROCEDURE Manage\_Network\_Information

INTERFACE ACCEPTS Object Identifiers, Community String and the Hostname or HostIP address

/\*Create the GetRequestPDU from the data accepted from the interface namely community string, hostname or the hostIP, object identifier. Send the GetResponsePDU by SNMP protocol on port 161.\*/

```
    Create_GetRequestPDU (CommStr, OID, hostIP);  
    Send_PDU (GetRequestPDU);
```

/\* Then it checks whether the community string is correct. It also checks whether the OID is present in the corresponding MIB. If the community string is incorrect or if the OID is not found in the MIB, it generates an error. If all parameters it retrieves the network information for the specified host. \*/

```
    Err1=Check_CommStr(CommStr);  
    If Err1= = true then  
        Display(" Invalid Community String ");  
        Exit(0);  
    End If
```

```
        Err2=Check_OID(OID);  
        If Err2= = true then  
            Display(" Invalid OID ");  
            Exit(0);  
        End If
```

```
    If Err1==false and Err2==false then  
        Rv=Check_type(CommStr);  
        If Rv==get  
            Call get_networkinfo();  
        Else If Rv==set  
            Call set_network_info();  
        End If  
    End If
```

/\* Then it creates a GetResponsePDU with the OID values and sends it back on port 161 using the SNMP protocol to the manager.\*/

```
    Create_GetResponsePDU (OID Values, Dest Port);  
    Send_PDU(GetResponsePDU);
```

**END PROCEDURE**

## 1.1 Manager View

<b>Attribute</b>	<b>Description</b>
Classification	Module
Definition	To provide manager view
Responsibilities	It is responsible for providing expert functions like selecting various OIDs and retrieving information about them, set values of OIDs , etc..
Constraints	The community string should be public and he should remember self-defined community strings.
Composition	-
Uses/Interactions	-
Resources	AdventNetLogging.jar, Applet Viewer, Browser
Processing	It retrieves the information about the network variables. It also sets the data for particular network variables. <b>Refer to PDL 1.1</b>
Interface/Exports	Displaydata : It displays the data related to the network variables Setdata : It sets the data related to the particular network variable.

### PDL 1.1

/\* The following Function retrieves the information about the specified host for the specified OID and returns an error if the object instance is not registered\*/

#### PROCEDURE get\_networkinfo()

SHELL ACCEPTS Object Identifiers, Community String and the Hostname or HostIP address

/\*The information about the specified OID is checked in the corresponding MIB for the corresponding host. If the object instance is unregistered, a No such object Instance found is returned. The presence of the host is checked. If the host is not present the request times out else the OID value of that host is returned. \*/

```

RetVal=Check_OIDreg(OID,hostIP);
If RetVal==false then
    Display(" No Such object Instance found ");
    Exit(0);
Else
    Hstprst== Check_Host(hostIP);
    If Hstprst==0 then
        Display("Request Timed Out");
        Exit(0);
    Else
        snmpget -c Commstr -v 2c hostIp OID
    End If
End If

```

**END PROCEDURE**

## 1. 2 User View

<b>Attribute</b>	<b>Description</b>
Classification	Module
Definition	To provide user view
Responsibilities	It is responsible for displaying the information about the network and check a particular OID value.
Constraints	-
Composition	-
Uses/Interactions	-
Resources	AdventNetLogging.jar, Applet Viewer, Browser
Processing	It retrieves the information about the network variables. <b>Refer to PDL 1.2</b>
Interface/Exports	Displaydata : It displays the data related to the network variables Setdata : It sets the data related to the particular network variable.

## PDL 1.2

/\* The following Function changes the information about the specified OID for the specified host and returns an error if the object instance is not registered or if the write permission is denied\*/

### PROCEDURE set\_networkinfo()

SHELL ACCEPTS Object Identifiers, Community String and the Hostname or HostIP address

/\*Firstly, the permission for changing the value of the OID is checked by checking the CommStr. If the CommStr is Public then the write permission is granted else it is denied.

Then the information about the specified OID is checked in the corresponding MIB for the corresponding host. If the object instance is unregistered, a No such object Instance found is returned. The presence of the host is checked. If the host is not present the request times out else the OID value of that host is returned.\*/

```
CommstrVal==Compare_CommStr(CommStr);
If CommstrVal== public then
  RetVal=Check_OIDreg(OID,hostIP);
  If RetVal==false then
    Display(" No Such object Instance found ");
    Exit(0);
  Else
    Hstprst== Check_Host(hostIP);
    If Hstprst==0 then
      Display("Request Timed Out");
      Exit(0);
    Else
      snmpset -c public -v 2c hostIp OID
    End If
  End If
Else
  Display ("Write Permission denied");
  Exit(0);
End If
```

**END PROCEDURE**

## 2. SNMP Agent:

<b>Attribute</b>	<b>Description</b>
Classification	Module
Definition	To provide facilities like managing the network information, monitoring network devices and extending MIB's and reporting changes to the SNMP Manager. It also processes the information requests sent by the Manager.
Responsibilities	It is responsible for providing facilities related to extending the MIB's, generating the code and header files for the MIB's, processing requests sent by the manager. It also does the task of sending the required information in the form of PDUs.
Constraints	Perl modules should be installed on the system.
Composition	MIB-header files : To provide the function definitions for functions used in code files MIB-Code files : To provide the functions to manage MIB objects LEAF-MIB : To define and initialize the objects required for LEAF
Uses/Interactions	It realizes the SNMP PDUs i.e. receives snmp PDUs and acts accordingly
Resources	Net-SNMP toolkit
Processing	It accepts the requests in the form of PDUs It parses the PDUs and checks for valid community strings and correct OID's. It sends the OID values in the form of PDUs and sends traps in the event of faults. <b>Refer to PDL 2</b>
Interface/Exports	Sendtrap : It sends a trap to the SNMP manager in the event of a fault. Senddata : It sends data to the SNMP manager while processing requests from the manager in the form of PDUs

## **PDL 2**

/\* The following procedure handles the management of the network MIB when a registration request is made by a new network device. During its execution it calls two procedures namely extend\_MIB() and generate\_MIBCode().

### **PROCEDURE** Manage\_Network\_MIB

INTERFACE ACCEPTS Object Identifiers, Configuration File

/\*Check for an incoming registration request from a network device. Accept the request from the device and send device info to the SNMP Manager. The Manager now defines the objects for the device according to ASN. He then registers instances of the objects and loads the MIB. The required C and Header files for the MIB are generated. Then the MIB is embedded in the agent.\*/\*

```
    Check_req(Reg_req);  
    Create_GetResponsePDU(Device Info);  
    Send_PDU(GetResponsePDU);
```

```
    Call extend_MIB();  
    Call generate_MIBCode();
```

**END PROCEDURE**

## 2.1 MIB Header Files:

<b>Attribute</b>	<b>Description</b>
Classification	Module
Definition	Provide the function declarations
Responsibilities	To provide the function declarartions for functions used in code files and supported data
Constraints	It is assumed that net-snmp toolkit is installed on the system and mib2c is functioning correctly. The files should be stored in the directory /agent/mibgroup/ and appropriate configuration file is used. <b>Refer to PDL 2.1</b>
Composition	-
Uses/Interactions	-
Resources	Net-SNMP toolkit
Processing	It provides the function declarations for the code files
Interface/Exports	-

## 2.2 MIB Code files:

<b>Attribute</b>	<b>Description</b>
Classification	Module
Definition	Provide the functions to manage MIB objects
Responsibilities	To provide the functions to manage MIB objects and initialize them and put the error messages in logs.
Constraints	It is assumed that net-snmp toolkit is installed on the system and mib2c is functioning correctly. The files should be stored in the directory /agent/mibgroup/ and appropriate configuration file is used.
Composition	-
Uses/Interactions	-
Resources	Net-SNMP toolkit
Processing	It registers the MIB and retrieves the value of the OID's. In the presence of errors it stores the error messages in logs. <b>Refer to PDL 2.2</b>
Interface/Exports	-

**PDL 2.1**

/\* The following Function generates the required C and header files for the MIB and embeds the agent with the MIB.\*/

**PROCEDURE** generate\_MIBCode()

SHELL Accepts Object Identifiers, Configuration File

/\*The presence of the object identifiers is checked. If the object identifier is absent then an Invalid OID error is generated. The daemon is reconfigured with the newly added module \*/

```
        Err4=Check_OID(OID);
    If Err4==true then
        Display("Invalid OID");
    Else
        env MIBS = ALL mib2c -c mib2c.mfd.conf OID
    End If
    ./configure --with --mibmodules="Module Identity"
    make
    make install
```

**END PROCEDURE**

### 2.3 LEAF-MIB:

<b>Attribute</b>	<b>Description</b>
Classification	Module
Definition	Define objects for LEAF
Responsibilities	To define the objects required for LEAF
Constraints	It should be defined in ASN.1 format
Composition	-
Uses/Interactions	-
Resources	Net-SNMP toolkit
Processing	It registers the instances of the objects It loads the default values of the objects and finally loads the MIB. <b>Refer to PDL 2.2</b>
Interface/Exports	-

#### **PDL 2.2**

/\* The following Function extends the MIB and loads the MIB. If the objects are wrongly defined the MIB is not loaded or object may not be found.\*/

#### **PROCEDURE** extend\_MIB()

/\*The objects for the device are defined by the SNMP Manager. The definition of the objects is checked. If the definition is proper the MIB is loaded and stored in the mibs directory else the MIB is not loaded. \*/

```
Define_objects(OIDs);
Err3==Check_MIBDef(MIB);
  If Err3== false then
    Reg_Instances(OIDs);
    Load_MIB(MIB);
    Store_MIB(MIG);
  Else
    Display ("MIB cannot be loaded");
    Exit(0);
  End If
```

**END PROCEDURE**

### 3. Network Information:

<b>Attribute</b>	<b>Description</b>
Classification	Interface
Definition	Provides network information
Responsibilities	To query the corresponding MIB's for the network variables and return their values
Constraints	-
Composition	-
Uses/Interactions	The Irpmodules interface is realized by this component through the use of various OID's
Resources	Net-SNMP toolkit
Processing	It queries the corresponding MIB for the specified OIDs It returns the values to the SNMP Agent.
Interface/Exports	ReplyPolling : It replies to the manager request SetOID : It sets the value for a particular OID reportValue : It reports the value of the particular OID to the manager in the form of PDU

### PDL 3

/\* This procedure handles the monitoring of network devices. During its execution it makes a call to two procedures namely Maintain\_NetworkTrans() and Maintain\_Access\_Policy().

#### PROCEDURE Monitor\_Network\_devices

INTERFACE ACCEPTS Community String, Host IP or Hostname

/\*Send a test packet to a network device. Process the response packet from the device and store log of the information. Create the GetRequestPDU from the data accepted from the interface namely community string, hostname or the hostIP, object identifier. Send the GetResponsePDU by SNMP protocol on port 161. The access rights are then checked and depending upon the access policies access is granted or denied.\*/

#### CALL Maintain\_NetworkTrans()

```
    Create_GetRequestPDU (CommStr, OID, hostIP);  
    Send_PDU (GetRequestPDU);
```

#### CALL Maintain\_Access\_Policy().

END PROCEDURE

#### 4. Package Information:

<b>Attribute</b>	<b>Description</b>
Classification	Module
Definition	Provides package information
Responsibilities	To retrieve the OID's related to package information
Constraints	-
Composition	-
Uses/Interactions	The Irmodules interface is realized by this component through the use of various OIDs
Resources	Net-SNMP toolkit
Processing	It queries the corresponding MIB for the specified OIDs It returns the values to the SNMP Agent.
Interface/Exports	-

#### PDL 3.1

/\* The following Function maintains the network transactions about the network devices and saves logs of the information. It also sends the status to the SNMP Manager. In case of failure a trap is sent to the Manager.\*/

#### PROCEDURE Maintain\_NetworkTrans()

/\*A test packet is sent to the device. The response packet is processed and the information is stored in form of logs and status is sent to the SNMP Manager. If the response packet doesn't return a time out occurs and a trap is sent to the manager.\*/

```
    Send_packet(Test_packet);
    Err3==accept_packet(Response_packet);
    If Err3== false then
        Process_Packet(Reponse_Packet);
        Store_log(Packet_Info);
        Create_GetResponsePDU(Status);
        Send_PDU(GetResponsePDU);
    Else
        Create_SetTrapPDU(Status);
        Send_PDU(SetTrapPDU);
```

End If

**END PROCEDURE**

## 5. System Information:

<b>Attribute</b>	<b>Description</b>
Classification	Module
Definition	Provides system information
Responsibilities	To query the corresponding MIB's for the system variables and return their values
Constraints	-
Composition	-
Uses/Interactions	The Irpmodules interface is realized by this component through the use of various OID's
Resources	Net-SNMP toolkit
Processing	It queries the corresponding MIB for the specified OIDs It returns the values to the SNMP Agent.
Interface/Exports	-

### **PDL 3.2**

/\* The following Function takes care of the access policy of the nodes in the network and the objects. In case of an access violation access is denied and error is generated else access is granted. \*/

#### **PROCEDURE Maintain\_Access\_Policy()**

SHELL ACCEPTS Community String and the Hostname or HostIP address

/\*The permission for changing the value of the OID is checked by checking the CommStr. If the CommStr is Public then the write permission is granted else it is denied. The permission for accessing the node is checked by checking the entries in the hosts.allow and hosts.deny file. If the hostIP is in the hosts.deny file the access should be denied else if it is in the hosts.allow file the access is granted.\*/

```
CommstrVal==Compare_CommStr(CommStr);

If CommstrVal== public then
    snmpset -c public -v 2c hostIp OID
    snmpget -c public -v 2c hostIp OID
Else
    Display ("Write Permission denied");
    Exit(0);
End If

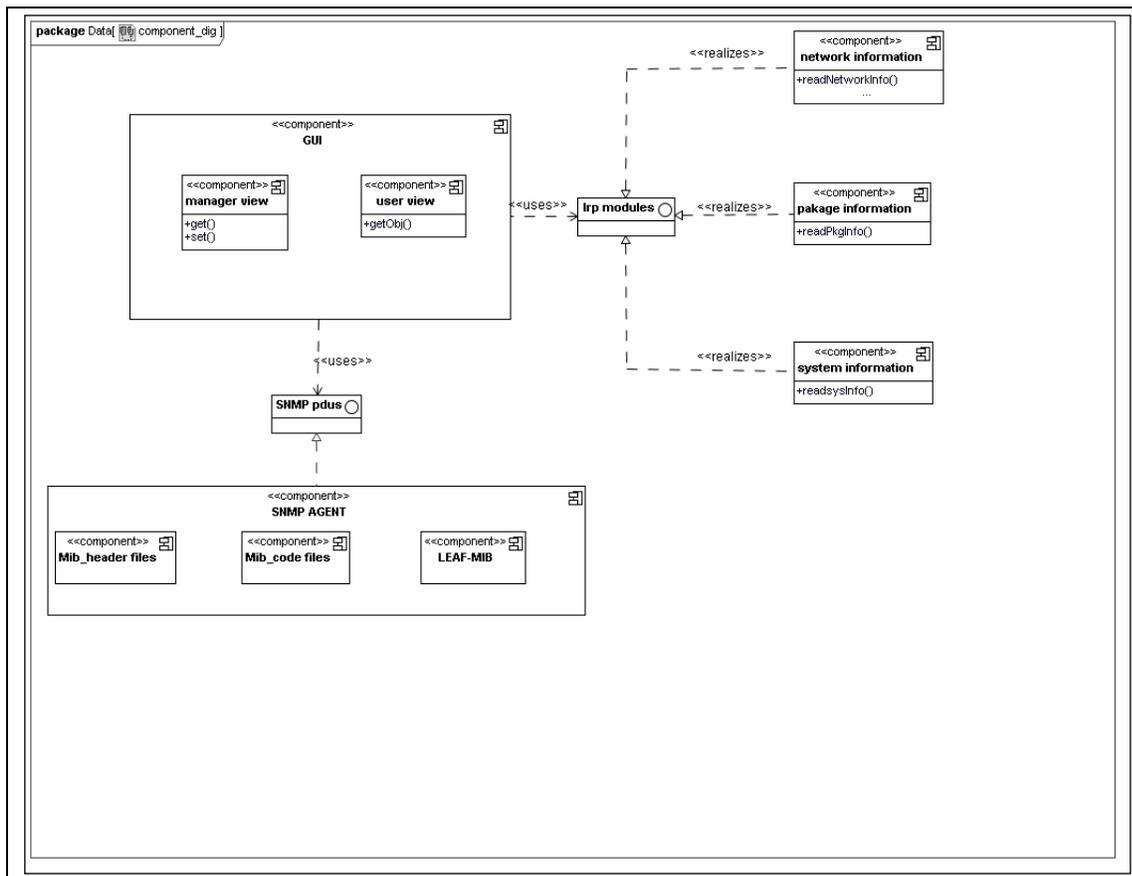
Fp=fopen(/etc/hosts.deny);
AccGrnt=Check_hostIP(HostIp,Fp);
If AccGrnt == false then
    Display("Access is Denied");
    Deny Access and terminate
Else
    Display("Access is Granted");
    Grant Access and continue
End If
End Procedure
```

### Interface Specification Template

<b>Interface</b>	SNMPPDU
<b>Description</b>	It provides connectivity between the SNMP Manager and the agent.
<b>Services</b>	Manage Network Information: It handles the retrieving and dispatching of network information. Monitor_Network_Devices: It handles the managing of access policies and managing transactions.
<b>Protocol</b>	The OID's need to be specified before the SNMP PDU is built
<b>Notes</b>	MIBCode Files, MIB Header Files and LEAF MIB.
<b>Issues</b>	-

### Interface Specification Template

<b>Interface</b>	Lrpmodules
<b>Description</b>	Brief description of the purpose of the interface.
<b>Services</b>	Manage_Network_MIB: It handles information related to the extension and loading of MIBs.
<b>Protocol</b>	The object instances need to be registered before the OID values are retrieved
<b>Notes</b>	Network information, Package information and System Information.
<b>Issues</b>	-



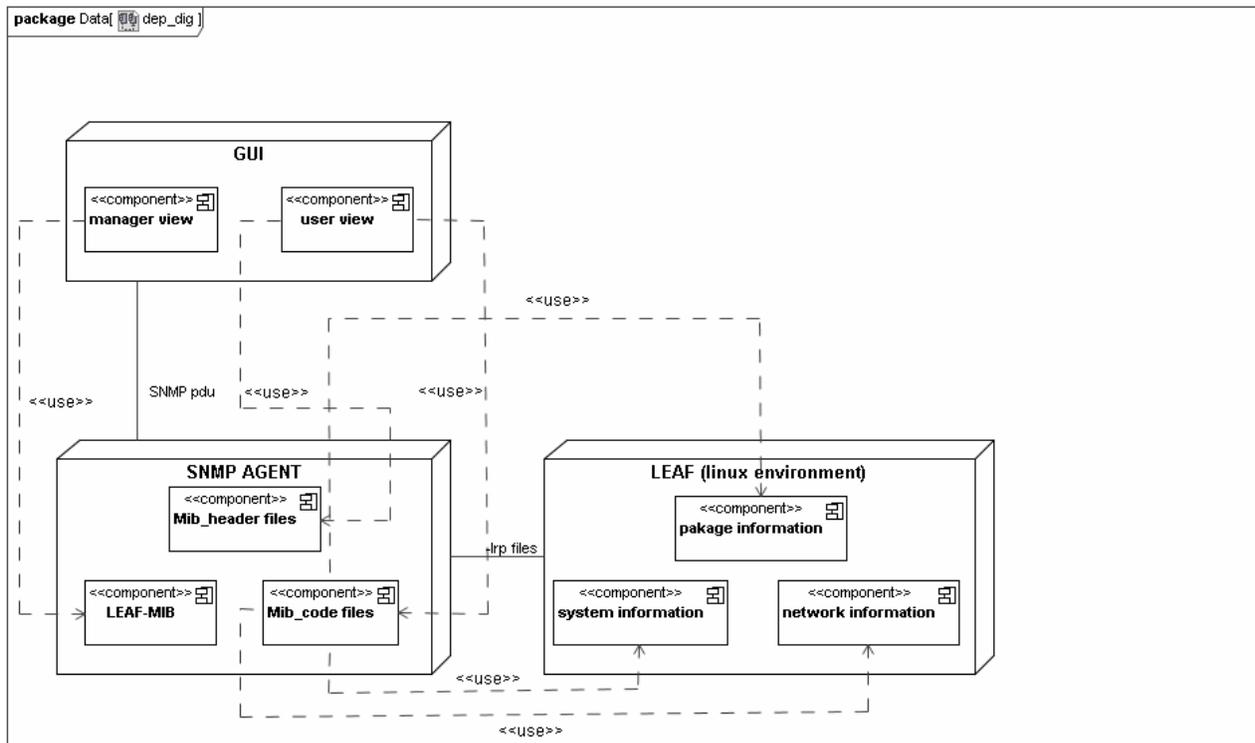
**Figure 10.1 Component Diagram**

## 2. DEPLOYMENT DIAGRAM DESCRIPTION

The deployment diagram consists of three main deployable items. First is Graphical user interface, second is SNMP Agent and lastly LEAF means Linux environment. User interface consists of Manager view who has advanced options and user view who has less privileges than manager.

The SNMP agent has LEAF-MIB, LEAF header file and LEAF code files. These files provide information to access object ids of objects present in LEAF-MIB. LEAF-MIB has various OIDs whose values show status of LEAF. The LEAF Linux environment provides files used by SNMP agent. It also provides information about system, about network and information about packets. The manager or user can get this information through Irp modules.

To start the system, one has to start SNMP agent and open GUI LogIn form. After authentication, depending on privilege level the person who has logged on can see and modify information in the OIDs.



**Figure 10.2 Deployment Diagram**

**SYMANTEC**

Project Title: LEAF-SNMP

# SYSTEM TESTING DOCUMENT

## TEST PLAN

### PURPOSE

The purpose of this test plan is to prescribe the scope, approach, resources, and schedule of the testing activities for this particular project. The aim of this document is to identify the items being tested, the features to be tested, and also the different testing tasks to be performed. The approaches we will be using for testing of the project are **Unit Testing** and **Integration Testing**.

Most part of the code and modules namely the **extended LEAF-MIB** is tested unit wise to ensure that the code produces exactly the same result as expected. It also helped to remove some bugs/errors that were bound to go undetected.

After each module/unit was tested and found working, they were integrated together and the final integration testing was performed. Integration Testing made sure that the system as a whole work as per specifications.

### Outline

The test plan has the following structure:

- a) Test plan identifier
- b) Introduction
- c) Test items
- d) Features to be tested
- e) Features not to be tested
- f) Item pass/fail criteria
- g) Environmental needs

## TEST PLAN IDENTIFIER

TP1

### INTRODUCTION

The software items to be tested are the goals that the software is designed to accomplish. These goals manifest themselves as functional requirements of the system. The following features of the system will be tested.

Goal : SNMP Agent should *monitor network devices* for network management.

Objectives :

- Maintain network transaction includes fault monitoring to maintain network channel information and report faults.
- Access policy includes providing proper node access and deciding access policies for non-registered SNMP managers.

Goal : SNMP Agent should *manage network information* for network management.

Objectives :

- Retrieve network information includes investigating network information and manipulating MIBs.
- Dispatch network information includes reply to manager polling and traps to manager on faults.

Goal : SNMP Agent should *manage Network\_MIB* for network management.

Objectives :

- Extend Management\_Info\_base includes defining objects according to ASN and registering those objects.
- Generate MIB\_Code includes generating C and header files to be modified as per requirement and embedding the MIB in the agent

### TEST ITEMS

The items that need to be tested include the retrieval and setting of the network parameters facilities which are handled by the **GUI**. The loading of the extended **LEAF MIB** also needs to be tested. Also, the **access policies** set for the network nodes need to be tested.

## FEATURES TO BE TESTED

The following features need to be tested:

- Loading of the extended MIB which is defined for the Linux Embedded Appliance Firewall (LEAF) needs to be tested.
- The access policies which are defined for the network nodes also need to be tested.
- The data retrieval facilities provided in the user and manager views needs to be tested. Also, the facility of setting the network variables in the manager view of GUI needs to be tested.
- Provide backward compatibility with the versions of SNMP namely SNMPv1 and SNMPv2 and also compatibility with future versions of the Linux Kernel.
- The data retrieval facilities should also be checked on the newly defined and loaded LEAF MIB objects. The community string also needs to be checked while handling the user and manager views in the GUI.
- Provide the facility for the handling of traps and displaying error messages in the event of traps in GUI.

These features directly map to the functional requirements of the system. Combinations of the above features need not be tested since all of them are generally used independently.

## FEATURES NOT TO BE TESTED

The features of the system that will not be tested are reliability, performance.

### 1.2.7 ITEM PASS/FAIL CRITERIA

Component	Pass Criteria	Fail Criteria
LEAF-MIB	MIB is loaded successfully and it returns OID values on request.	MIB does not load due to wrong definition of the objects.
GUI	Values of the network parameters are retrieved successfully.	Unable to access the network parameters or Request timed out.
Security	Nodes who are banned in the hosts.deny file do not get access.	Node is banned to access the network and it can access network parameters.

## **ENVIRONMENTAL NEEDS**

Environmental needs include

- Linux Distribution (RedHat, OpenSuse 10.2,etc) to be present on the system
- Net-SNMP toolkit to be installed on the Linux Distribution
- Java run time environment installed to be installed on the Linux Distribution
- AdventNetLogging.jar to be present on the system
- Browser supporting Appletviewer to be present on the system

## TEST CASE SPECIFICATION

### 1) Use Case : Monitor Network Devices

<b>Test Case Identifier</b>	Monitor network devices
<b>Test Items</b>	Maintain network transaction. Maintain Access policy.
<b>Input Specifications</b>	Community String::public, IP Address::127.0.0.1 or Agents IP
<b>Output Specifications</b>	Object values::system.sysName.0=Linux
<b>Other Intercase Dependencies</b>	Nil
Successful execution of Monitor Network Device leads to reporting of traps in case of faults, storing of logs and no access violations.	

<b>Test Procedure Identifier</b>	Monitor network devices
<b>Purpose</b>	To test the monitoring of the network devices according to the management of network transactions and access policies which are defined for the network.
<b>Test Procedure</b>	<pre> /* This procedure handles the monitoring of network devices. During its execution it makes a call to two procedures namely Maintain_NetworkTrans() and Maintain_Access_Policy().  <b>PROCEDURE</b> Monitor_Network_devices  INTERFACE ACCEPTS Community String, Host IP or Hostname  /*Send a test packet to a network device. Process the response packet from the device and store log of the information. Create the GetRequestPDU from the data accepted from the interface namely community string, hostname or the hostIP, object identifier. Send the GetResponsePDU by SNMP protocol on port 161. The access rights are then checked and depending upon the access policies access is granted or denied.*/  <b>CALL</b> Maintain_NetworkTrans()  Create_GetRequestPDU (CommStr, OID, hostIP); Send_PDU (GetRequestPDU);  <b>CALL</b> Maintain_Access_Policy().  <b>END PROCEDURE</b> </pre>
<b>Procedure Result</b>	Network transactions are maintained and access policy is maintained.
<b>Anomalous Events</b>	Violation of access rights, Unhandled fault.

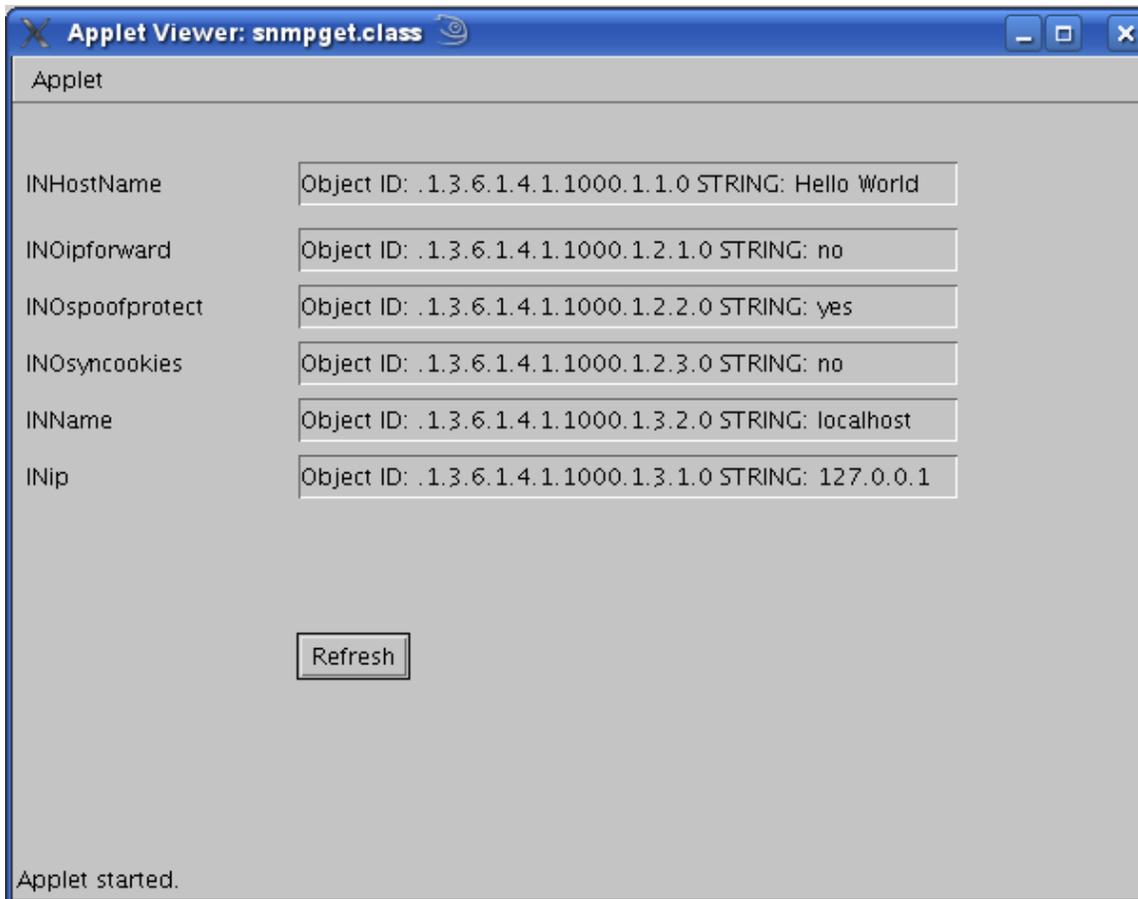


Figure 11.1 Snapshot :: Monitor Network Devices

### 1.1. Use Case : Monitor Network Transaction

<b>Test Case Identifier</b>	Maintain network transaction
<b>Test Items</b>	SNMP PDUs
<b>Input Specifications</b>	Community String::public, IP Address::127.0.0.1 or Agent's IP address
<b>Output Specifications</b>	Trap sent to the manager and log file is maintained
<b>Other Intercase Dependencies</b>	Nil
Successful execution of Monitor Network Device leads to reporting of traps in case of faults, storing of logs and no access violations.	

<b>Test Procedure Identifier</b>	Maintain network transaction
<b>Purpose</b>	To test whether network transaction is handled correctly by maintaining of logs and sending of traps in the event of faults.
<b>Test Procedure</b>	<p><b>PROCEDURE</b> Maintain_NetworkTrans()</p> <p>/*A test packet is sent to the device. The response packet is processed and the information is stored in form of logs and status is sent to the SNMP Manager. If the response packet doesn't return a time out occurs and a trap is sent to the manager.*/</p> <p>Send_packet(Test_packet); Err3==accept_packet(Response_packet);</p> <p>If Err3== false then  <b>Process_Packet(Reponse_Packet);</b>  <b>Store_log(Packet_Info);</b>  <b>Create_GetResponsePDU(Status);</b>  <b>Send_PDU(GetResponsePDU);</b>  Else  <b>Create_SetTrapPDU(Status);</b>  <b>Send_PDU(SetTrapPDU);</b>  End If</p> <p><b>END PROCEDURE</b></p>
<b>Procedure Result</b>	Network transaction logs are maintained and traps are sent in the event of faults.
<b>Anomalous Events</b>	Unhandled Fault.

```

config.log - KWrite
File Edit View Bookmarks Tools Settings Help
configure:4284: $? = 1
configure: failed program was:
| #ifndef __cplusplus
|   choke me
| #endif
configure:4422: checking how to run the C preprocessor
configure:4457: gcc -E conftest.c
configure:4463: $? = 0
configure:4495: gcc -E conftest.c
conftest.c:15:28: error: ac_nonexistent.h: No such file or directory
configure:4501: $? = 1
configure: failed program was:
| /* confdefs.h.  */
| .
| #define PACKAGE_NAME "Net-SNMP"
| #define PACKAGE_TARNAME "net-snmp"
| #define PACKAGE_VERSION "5.4.1"
| #define PACKAGE_STRING "Net-SNMP 5.4.1"
| #define PACKAGE_BUGREPORT "net-snmp-coders@lists.sourceforge.net"
| #define NETSNMP_CONFIGURE_OPTIONS '-with-mib-modules=leafNetwork'
| #define NETSNMP_ENABLE_SCAPI_AUTHPRIV 1
| #define NETSNMP_PERSISTENT_MASK 077
| #define NETSNMP_AGENTX_SOCKET "/var/agentx/master"
| #define NETSNMP_NO_DUMMY_VALUES 1
| #define NETSNMP_WITH_OPAQUE_SPECIAL_TYPES 1
| /* end confdefs.h.  */
| #include <ac_nonexistent.h>
configure:4540: result: gcc -E
configure:4564: gcc -E conftest.c
configure:4570: $? = 0
configure:4602: gcc -E conftest.c
conftest.c:15:28: error: ac_nonexistent.h: No such file or directory
configure:4608: $? = 1
configure: failed program was:
| /* confdefs.h.  */
| .
| #define PACKAGE_NAME "Net-SNMP"
| #define PACKAGE_TARNAME "net-snmp"

```

Figure 11.2 Snapshot :: Maintain Network Transaction

## 1.2. Use Case : Maintain Access Policy

<b>Test Case Identifier</b>	Maintain Access policy.
<b>Test Items</b>	Hosts.deny file, Community String
<b>Input Specifications</b>	Request for node access or specifying a community string
<b>Output Specifications</b>	Access is denied or invalid community string gets rejected.
<b>Other Intercase Dependencies</b>	Nil
Successful execution of Maintain Access policy leads to the prevention of invalid accesses to nodes or unregistered community strings are rejected.	

<b>Test Procedure Identifier</b>	Maintain Access Policy
<b>Purpose</b>	To test the access policy defined for the users and the network.
<b>Test Procedure</b>	<pre> /* The following Function takes care of the access policy of the nodes in the network and the objects. In case of an access violation access is denied and error is generated else access is granted. */ <b>PROCEDURE</b> Maintain_Access_Policy()  SHELL ACCEPTS Community String and the Hostname or HostIP address /*The permission for changing the value of the OID is checked by checking the CommStr. If the CommStr is Public then the write permission is granted else it is denied. The permission for accessing the node is checked by checking the entries in the hosts.allow and hosts.deny file. If the hostIP is in the hosts.deny file the access should be denied else if it is in the hosts.allow file the access is granted.*/ CommstrVal==Compare_CommStr(CommStr); If CommstrVal== public then     snmpset -c public -v 2c hostIp OID     snmpget -c public -v 2c hostIp OID Else     Display (“Write Permission denied”);     Exit(0); End If Fp=fopen(/etc/hosts.deny); AccGrnt=<b>Check_hostIP(HostIp,Fp)</b>;     If AccGrnt == false then         Display(“Access is Denied”);         Deny Access and terminate     Else         Display(“Access is Granted”);         Grant Access and continue     End If <b>END PROCEDURE</b> </pre>
<b>Procedure Result</b>	The privilege levels are assigned to users and access to OIDs can be banned to some users.
<b>Anomalous Events</b>	Write permission denied, request timed out

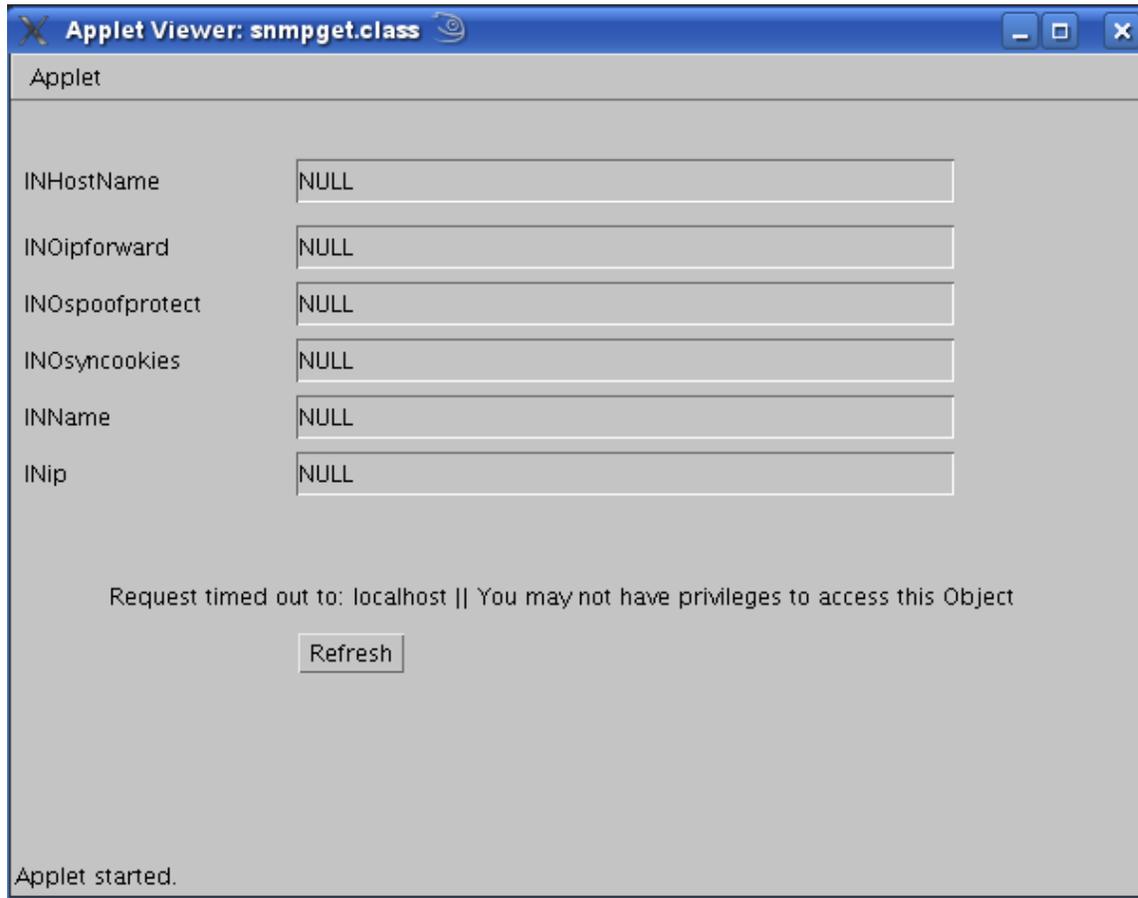


Figure 11.3 Snapshot :: Maintain Access Policy

## 2. Use Case : Manage Network Information

<b>Test Case Identifier</b>	Manage network information
<b>Test Items</b>	Retrieve network information. Dispatch network information
<b>Input Specifications</b>	Community String::public , IP Address::127.0.0.1 or Agents IP address, OID : LEAF-MIB::INHostName.0
<b>Output Specifications</b>	Object values:: Name of LEAF Network host
<b>Other Intercase Dependencies</b>	Nil
Successful execution of Manage network information leads to successful retrieval and setting of values of the specifies OIDs.	

<b>Test Procedure Identifier</b>	Manage Network Information
<b>Purpose</b>	To test the handling of the network information is managed properly.
<b>Test Procedure</b>	<pre> /* The following procedure handles the management of the network information when a request is made by the SNMP Manager. During its execution it calls two procedures namely get_networkinfo() and set_networkinfo().*/  <b>PROCEDURE</b> Manage_Network_Information  INTERFACE ACCEPTS Object Identifiers, Community String and the Hostname or HostIP address  /*Create the GetRequestPDU from the data accepted from the interface namely community string, hostname or the hostIP, object identifier. Send the GetResponsePDU by SNMP protocol on port 161.*/  Create_GetRequestPDU (CommStr, OID, hostIP); Send_PDU (GetRequestPDU);  /* Then it checks whether the community string is correct. It also checks whether the OID is present in the corresponding MIB. If the community string is incorrect or if the OID is not found in the MIB, it generates an error. If all parameters it retrieves the network information for the specified host. */                  Err1=Check_CommStr(CommStr); If Err1= = true then     Display(" Invalid Community String ");     Exit(0); End If  Err2=Check_OID(OID); If Err2= = true then     Display(" Invalid OID ");     Exit(0); End If  If Err1==false and Err2==false then     Rv=Check_type(CommStr);     If Rv==get </pre>

	<p>Call <b>get_networkinfo()</b>;  Else If Rv==set  Call <b>set_network_info()</b>;  End If  End If</p> <p>/* Then it creates a GetResponsePDU with the OID values and sends it back on port 161 using the SNMP protocol to the manager.*/  <b>Create_GetResponsePDU (OID Values, Dest Port);</b>  <b>Send_PDU(GetResponsePDU);</b></p> <p><b>END PROCEDURE</b></p>
<b>Procedure Result</b>	The values of network parameters are retrieved and also can be set by the SNMP Manager.
<b>Anomalous Events</b>	Invalid OIDs, write permission denied, request timed out and object instance not found.

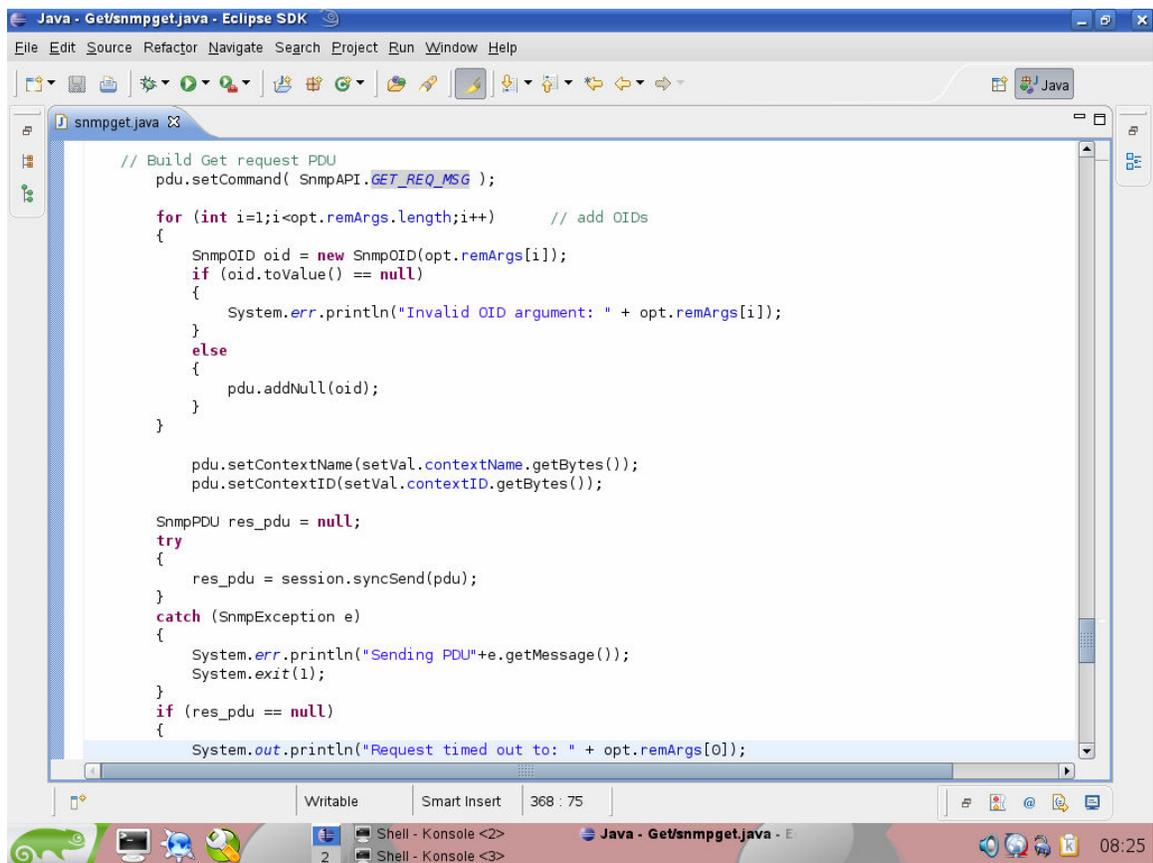


Figure 11.4 Snapshot :: Manage Network Information

## 2.1. Use Case : Retrieve Network Information

<b>Test Case Identifier</b>	Retrieve network information
<b>Test Items</b>	Object Identifiers
<b>Input Specifications</b>	Community String::public , IP Address::127.0.0.1 or Agents IP address, OID : LEAF-MIB::INHostName.0
<b>Output Specifications</b>	Object values:: Name of LEAF Network
<b>Other Intercase Dependencies</b>	Nil
Successful execution of Retrieve network information leads to successful retrieval of values of the specifies OIDs.	

<b>Test Procedure Identifier</b>	Retrieve Network Information
<b>Purpose</b>	To test whether the values of the network parameters are retrieved successfully.
<b>Test Procedure</b>	<pre> /* The following Function retrieves the information about the specified host for the specified OID and returns an error if the object instance is not registered*/  <b>PROCEDURE</b> get_networkinfo()  SHELL ACCEPTS Object Identifiers, Community String and the Hostname or HostIP address  /*The information about the specified OID is checked in the corresponding MIB for the corresponding host. If the object instance is unregistered, a No such object Instance found is returned. The presence of the host is checked. If the host is not present the request times out else the OID value of that host is returned. */  RetVal=Check_OIDreg(OID,hostIP); If RetVal==false then   Display(" No Such object Instance found ");   Exit(0); Else   Hstprst== Check_Host(hostIP);   If Hstprst==0 then     Display("Request Timed Out");     Exit(0);   Else     <b>snmpget -c Commstr -v 2c hostIp OID</b>   End If End If <b>END PROCEDURE</b> </pre>
<b>Procedure Result</b>	The values of network parameters are retrieved successfully.
<b>Anomalous Events</b>	Invalid OIDs, request timed out and object instance not found.

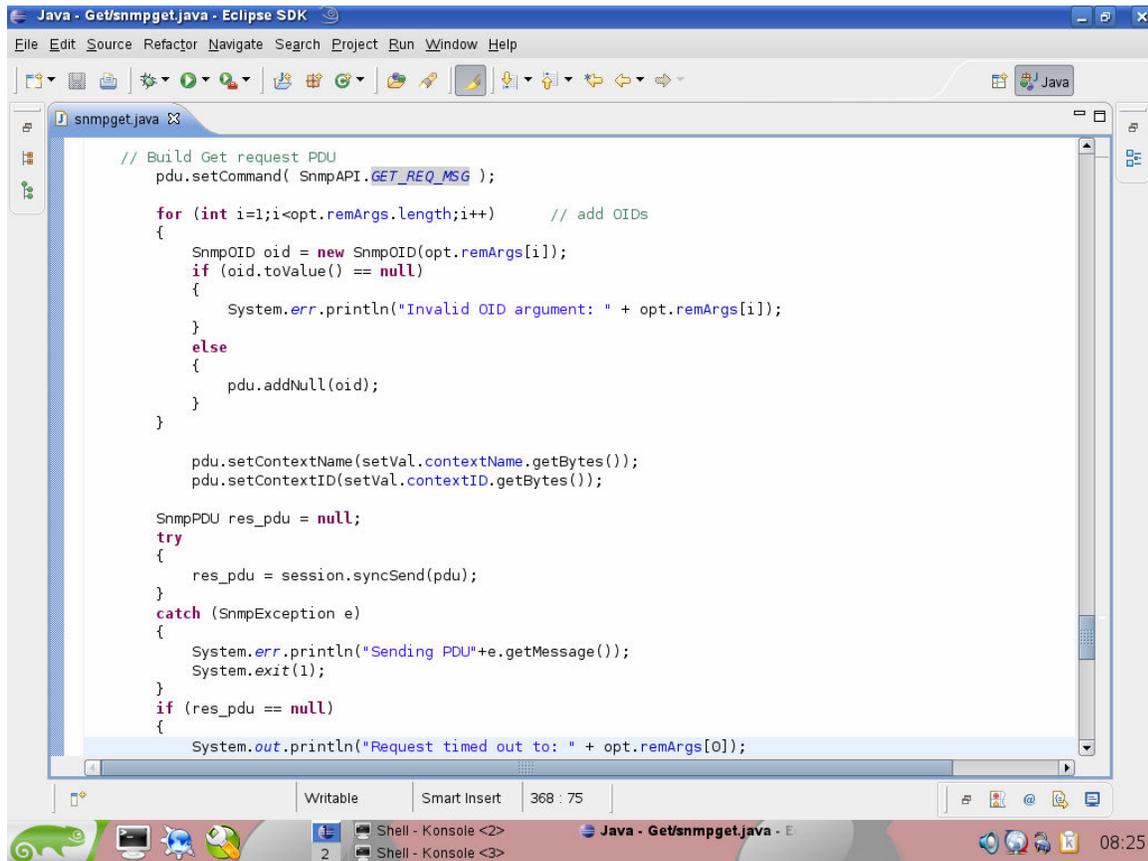
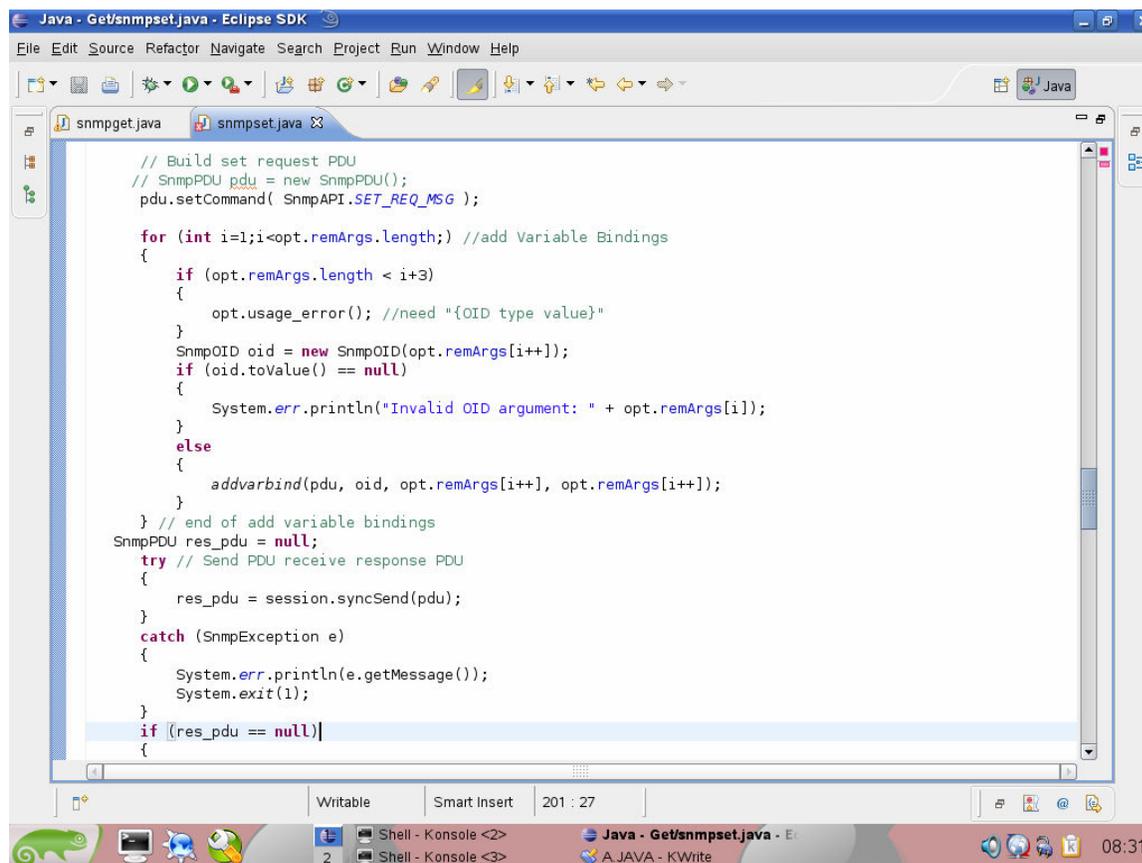


Figure 11.5 Snapshot :: Retrieve Network Information

## 2.2. Use Case : Dispatch Network Information

<b>Test Case Identifier</b>	Dispatch network information
<b>Test Items</b>	Object Identifiers
<b>Input Specifications</b>	Community String::public , IP Address::127.0.0.1 or Agents IP address, OID : LEAF-MIB::INHostName.0
<b>Output Specifications</b>	Object values:: New name of LEAF Network
<b>Other Intercase Dependencies</b>	Nil
Successful execution of Dispatch network information leads to successful setting of values of the specifies OIDs.	

<b>Test Procedure Identifier</b>	Dispatch Network Information
<b>Purpose</b>	To test whether the values of the network parameters are set correctly.
<b>Test Procedure</b>	<pre> /* The following Function changes the information about the specified OID for the specified host and returns an error if the object instance is not registered or if the write permission is denied*/  <b>PROCEDURE</b> set_networkinfo() SHELL ACCEPTS Object Identifiers, Community String and the Hostname or HostIP address /*Firstly, the permission for changing the value of the OID is checked by checking the CommStr. If the CommStr is Public then the write permission is granted else it is denied. Then the information about the specified OID is checked in the corresponding MIB for the corresponding host. If the object instance is unregistered, a No such object Instance found is returned. The presence of the host is checked. If the host is not present the request times out else the OID value of that host is returned.*/ CommstrVal==Compare_CommStr(CommStr); If CommstrVal== public then    RetVal=Check_OIDreg(OID,hostIP); If RetVal==false then     Display(“ No Such object Instance found “);     Exit(0); Else     Hstprst== Check_Host(hostIP); If Hstprst==0 then     Display(“Request Timed Out”);     Exit(0); Else     <b>snmpset -c public -v 2c hostIp OID</b> End If End If Else     Display (“Write Permission denied”);     Exit(0); End If <b>END PROCEDURE</b> </pre>
<b>Procedure Result</b>	The values of network parameters are set by the SNMP Manager successfully.
<b>Anomalous Events</b>	Invalid OIDs, write permission denied, request timed out and object instance not found.

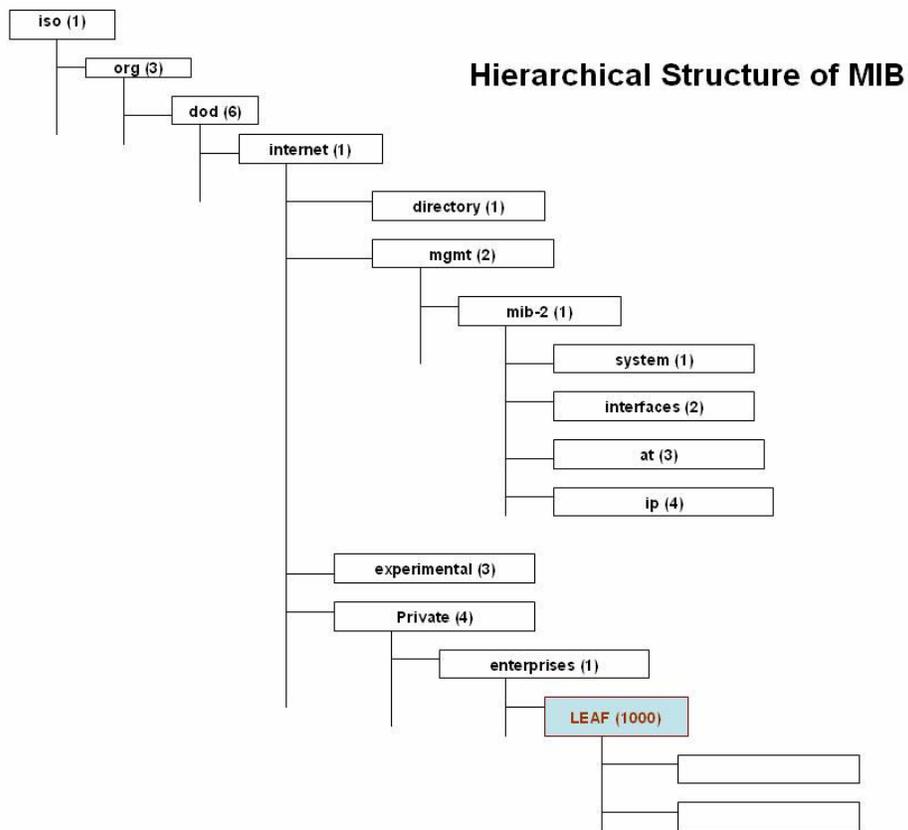


**Figure 11.6 Snapshot :: Dispatch Network Information**

### 3. Use Case : Manage Network MIB

<b>Test Case Identifier</b>	Manage Network_MIB
<b>Test Items</b>	Extend Management_Info_Base. Generate MIB_Code
<b>Input Specifications</b>	MIB written in ASN.1 format
<b>Output Specifications</b>	Loaded MIB
<b>Other Intercase Dependencies</b>	Nil
Successful execution of Manage Network_MIB leads to successful generation of the MIB code files and loading of MIB.It also leads to the MIB getting embedded in the agent.	

<b>Test Procedure Identifier</b>	Manage Network_MIB
<b>Purpose</b>	To test whether the MIB is extended correctly and the Code and header files for the MIB are generated. Also, test whether the object instances are registered and MIB is embedded into the agent.
<b>Test Procedure</b>	<pre> /* The following procedure handles the management of the network MIB when a registration request is made by a new network device. During its execution it calls two procedures namely extend_MIB() and generate_MIBCode().  <b>PROCEDURE</b> Manage_Network_MIB  INTERFACE ACCEPTS Object Identifiers, Configuration File  /*Check for an incoming registration request from a network device. Accept the request from the device and send device info to the SNMP Manager. The Manager now defines the objects for the device according to ASN. He then registers instances of the objects and loads the MIB. The required C and Header files for the MIB are generated. Then the MIB is embedded in the agent.*/  Check_req(Reg_req); Create_GetResponsePDU(Device Info); Send_PDU(GetResponsePDU);  Call <b>extend_MIB()</b>; Call <b>generate_MIBCode()</b>;  <b>END PROCEDURE</b> </pre>
<b>Procedure Result</b>	Network MIB is loaded successfully and embedded into the SNMP agent and the object instances are registered successfully.
<b>Anomalous Events</b>	MIB not loaded, Segmentation fault

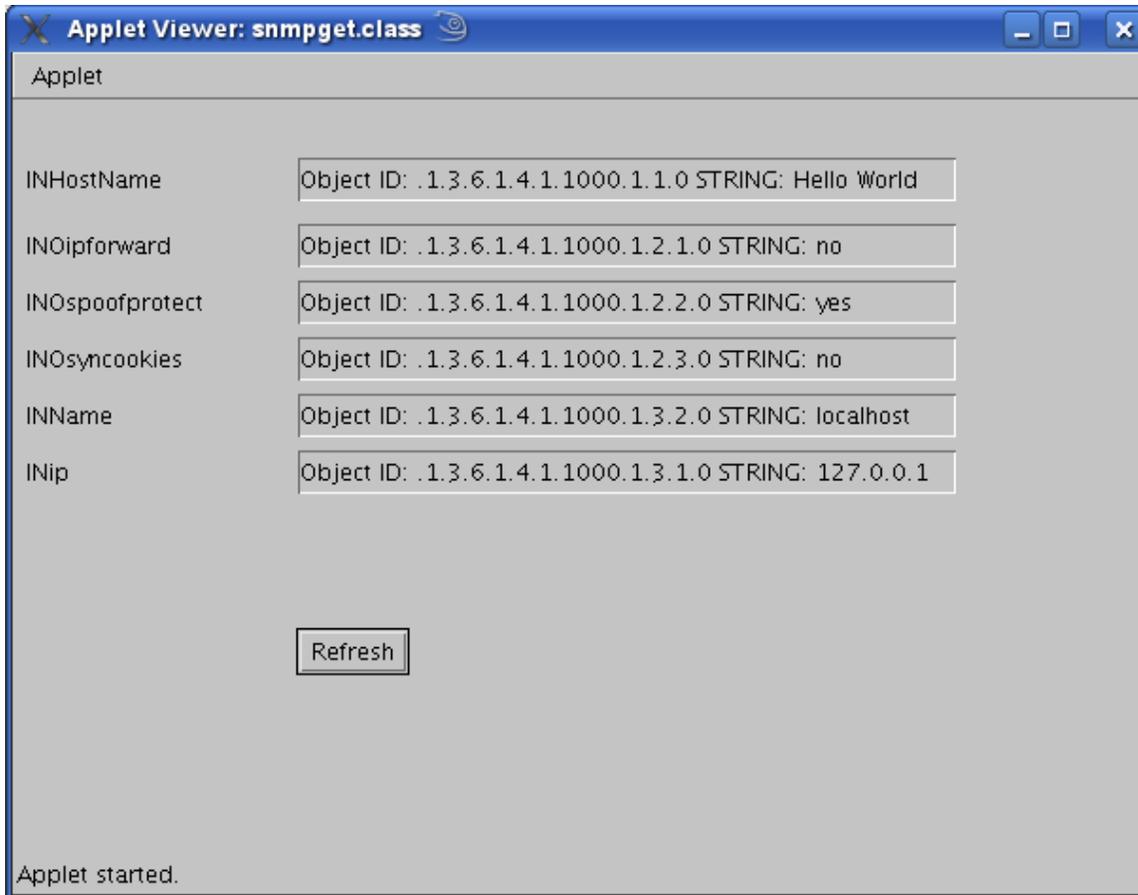


**Figure 11.7 Snapshot :: Manage Network MIB**

### 3.1 Use Case : Extend MIB

<b>Test Case Identifier</b>	Extend Management_Info_Base.
<b>Test Items</b>	Objects defined in the MIB
<b>Input Specifications</b>	MIB written in ASN.1 format
<b>Output Specifications</b>	Registered instances of objects defined in the MIB.
<b>Other Intercase Dependencies</b>	Nil
Successful execution of Extend Management_Info_Base. leads to registering of instances of the objects defined in the MIB.	

<b>Test Procedure Identifier</b>	Extend Management_Info_Base
<b>Purpose</b>	To test whether the MIB is extended correctly and the object instances are registered successfully
<b>Test Procedure</b>	<pre> /* The following Function extends the MIB and loads the MIB. If the objects are wrongly defined the MIB is not loaded or object may not be found.*/  <b>PROCEDURE</b> extend_MIB()  /*The objects for the device are defined by the SNMP Manager. The definition of the objects is checked. If the definition is proper the MIB is loaded and stored in the mibs directory else the MIB is not loaded. */  Define_objects(OIDs);  Err3==Check_MIBDef(MIB); If Err3== false then   <b>Reg_Instances(OIDs);</b>   <b>Load_MIB(MIB);</b>   <b>Store_MIB(MIG);</b> Else   Display (“MIB cannot be loaded”);   Exit(0); End If  <b>END PROCEDURE</b> </pre>
<b>Procedure Result</b>	Object instances of new MIB are registered.
<b>Anomalous Events</b>	MIB cannot be loaded, Object not found.



**Figure 11.8 Snapshot :: Extends MIBs**

### 3.2 Use Case : Generate MIB\_Code

<b>Test Case Identifier</b>	Generate MIB_Code
<b>Test Items</b>	MIB
<b>Input Specifications</b>	MIB written in ASN.1 format
<b>Output Specifications</b>	To Code and header files generated for the MIB and loading of the MIB
<b>Other Intercase Dependencies</b>	Nil
Successful execution of Generate MIB_Code leads to generation of code and header files and successful loading of the MIB and embedding in the agent.	

<b>Test Procedure Identifier</b>	Generate MIB_Code
<b>Purpose</b>	To test whether the code and header files for MIB are generated successfully and the MIB is embedded into the agent.
<b>Test Procedure</b>	<pre> /* The following Function generates the required C and header files for the MIB and embeds the agent with the MIB.*/  <b>PROCEDURE</b> generate_MIBCode()  SHELL Accepts Object Identifiers, Configuration File  /*The presence of the object identifiers is checked. If the object identifier is absent then an Invalid OID error is generated. The daemon is reconfigured with the newly added module */  Err4=Check_OID(OID); If Err4==true then   Display("Invalid OID"); Else <b>env MIBS = ALL mib2c -c mib2c.mfd.conf</b> <b>OID</b> End If . ./configure --with --mibmodules="Module Identity" <b>make</b> <b>make install</b>  <b>END PROCEDURE</b> </pre>
<b>Procedure Result</b>	Code and header files for MIB are generated and MIB is embedded in the SNMP agent.
<b>Anomalous Events</b>	Invalid OID, Private MIB cannot be loaded

```
leafSystem.c - KWrite
File Edit View Bookmarks Tools Settings Help
*
* Please see the documentation for more information about writing
* module extensions, and check out the examples in the examples
* and mibII directories.
*/
unsigned char *
var_leafSystem(struct variable *vp,
               oid *name,
               size_t *length,
               int exact,
               size_t *var_len,
               WriteMethod **write_method)
{
    /* variables we may use later */
    static long long_ret;

    if (header_generic(vp,name,length,exact,var_len,write_method)
        == MATCH_FAILED )
        return NULL;

    /*
     * this is where we do the value assignments for the mib results.
     */
    switch(vp->magic) {
        case LSLLOGSDAILY:
            *write_method = write_lsllogsdaily;
            *var_len = lsllogsdaily_len;
            return (unsigned char *) lsllogsdaily;

        case LSLLOGSWEEKLY:
            *write_method = write_lsllogsweekly;
            *var_len = lsllogsweekly_len;
            return (unsigned char *) lsllogsweekly;

        case LSLLOGSMONTHLY:
            *write_method = write_lsllogsmonthly;
            *var_len = lsllogsmonthly_len;
            return (unsigned char *) lsllogsmonthly;
    }
}
```

Figure 11.9 Snapshot :: Generate MIB Code

## CONCLUSION

The goal of this work was to extend an SNMP agent which works in a LEAF (Linux Embedded Appliance Firewall) emulated Linux environment. This product is working for network management in form of Fault, Configuration, Accounting, Performance, and Security. This agent extends MIB which acts as information base for above tasks. Currently SNMP agents are present for kernel version 1.0 and less. The agent that is developed as part of the project works for kernel version 2.6 and above.. Currently existing agent needs all commands to be typed by user on console which is very complicated task. It provides facility to extend MIBs as per requirements. The LEAF-MIB has been defined and used to extend existing agent, strengthening security for the network.

The clear differentiation between existing system and the developed agent can be give as follows:

Existing system :

Merits:

- Handles Faults, security, accounting, configuration, security
- Allows to retrieve and set values in instances of managed objects
- Can be managed by any SNMP manager

Demerits:

- Lacks a GUI
- Security implementations are inefficient

Developed system:

Merits:

- Robust GUI
- Extension of MIB (e.g. LEAF-MIB)
- Security issues of previous versions are handled
- Integration of existing agent in LEAF

Demerits:

- Cannot be handled by all the SNMP managers
- Performance issue not handled
- Trap issue is not handled satisfactorily

Finally, this SNMP agent extends MIB for Linux Embedded Appliance Firewall and provides simple GUI to retrieve and set network information. Thus this system fills the gap between what is existing and what is required for network management.

## REFERENCES

1. **SNMP, SNMPv2, SNMPv3, and RMON 1 and 2 3rd Edition**  
William Stallings  
Addison-Wesley, 1998
2. **LEAF Documentation**  
Bering-uClibC Team, March 2007
3. **Net-SNMP Documentation & Wiki**  
Net-SNMP Team, March 2007  
<http://en.wikipedia.org/wiki/Net-SNMP>
4. **CISCO PIX Firewall**  
[http://en.wikipedia.org/wiki/Cisco\\_PIX](http://en.wikipedia.org/wiki/Cisco_PIX)
5. **Net-SNMP and LEAF Mailing lists**

