MIB VALUE READER

P-671

PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING

OF BHARATHIAR UNIVERSITY,

COIMBATORE.

Submitted by

SRIKANTH P.S.
RAMACHANDRAN B.
PRIYA ALPHONSE

Guided by

Mrs S. DEVAKI M.S.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE 641 006

MARCH 2002

KUMARAGURU COLLEGE OF TECHNOLOGY

(Affiliated to Bharathiar University, Coimbatore)

CERTIFICATE

This is to certify that project report entitled

MIB VALUE READER

Is a bonafide record of work done by

SRIKANTH P. S.	**9827KO217**
RAMACHANDRAN B.	**9827KO204**
PRIYA ALPHONSE	**9827KO200**

IN PARTIAL FULFUILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING

Head Of the Department

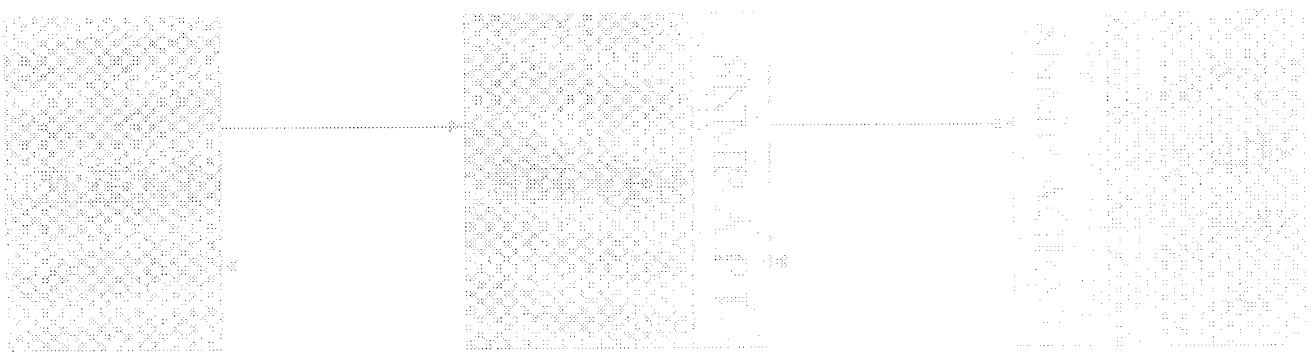
Staff-in-charge

Submitted for the University Examination held on _____

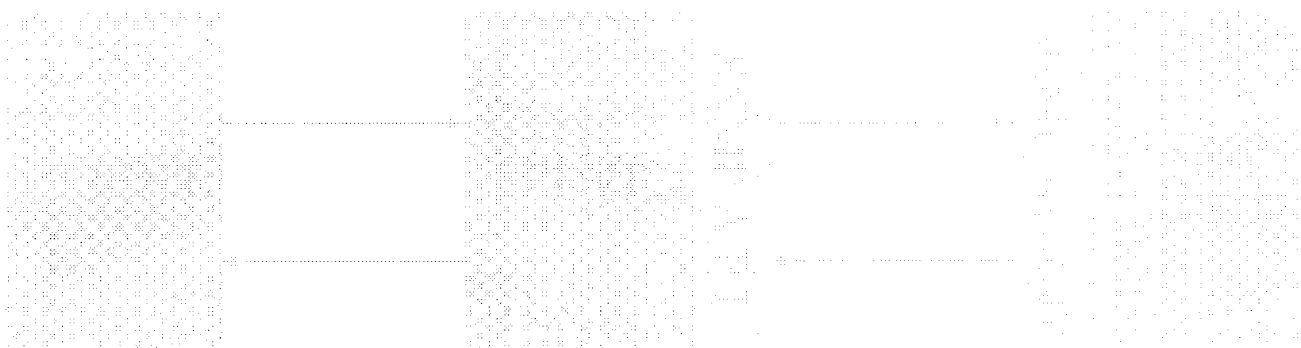
Internal Examiner

External Examiner

Place: Coimbatore
Date:



We dedicate this project to our parents and friends.



ACKNOWLEDGEMENT

Ever since the day we came up with the idea behind our project, we have been fortunate to have had the support and goodwill of several people who matter very much to us. Our gratitude to all of them is immense and we try here to express them as best as we can.

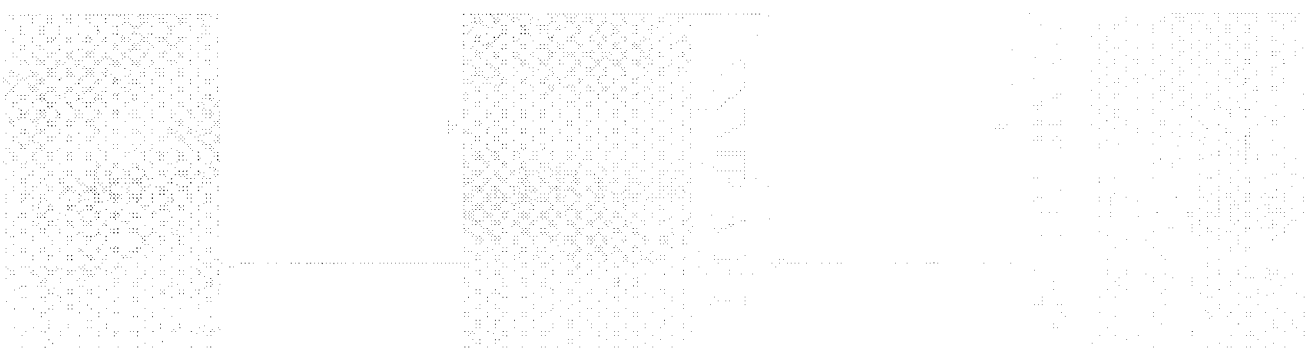
We express our sincere gratitude to **Dr. K. K. Padmanabhan**, our esteemed principal, for giving us the needed encouragement in carrying out this project successfully.

Prof. S. Thangaswamy, the head of the department of Computer Science and Engineering is a person of immense knowledge and his suggestions have been very useful to us in the course of development of this project and we are indebted to you sir.

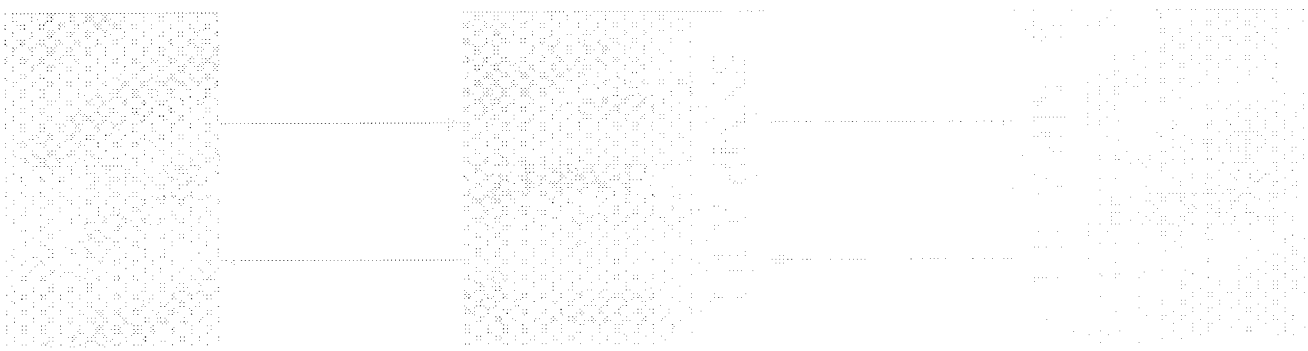
We thank our guide **Asst. Prof. Mrs. S. Devaki**, Department of Computer Science and Engineering whose tremendous motivation and guidance enabled us to embark on a project of this magnitude. We are grateful for all the kindness shown to us by all our teaching and non-teaching staff, especially our lab technicians, who have had late lunches on many, an occasion because of helping us in the lab.

Our project guide **Mr. Prasuj Loganathan** has always gone out of his way in spite of his ever so hectic schedule to help us out throughout the project development and we hold precious our association with him.

Our families have been very understanding and supportive in putting up with our late hours of work and total exclusion from them all for a couple of weeks at a time. We are incredibly lucky to have such wonderful families. To all our friends and acquaintances who have helped us in any measure, thanks folks.



CONTENTS



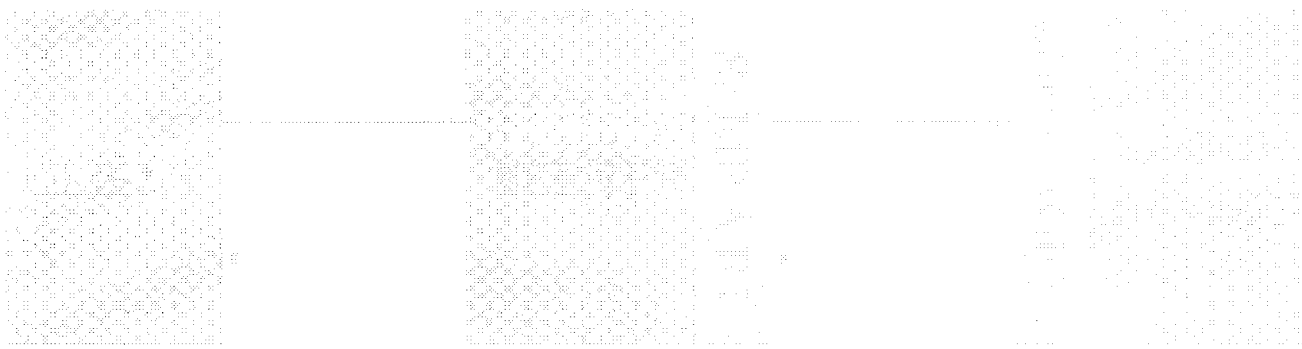
SYNOPSIS

The goal of our project titled "MIB Value Reader" is to read run time values of object identifiers which are contained in the MIB files of various SNMP compatible devices that are connected in the network. Our project implements the concepts of Remote Method Invocation (RMI) and Simple Network Management Protocol (SNMP).

Our system is defined by three participants: the RMI server, the client that runs the client application and the device(s) in the network that are monitored.

The client application allows the network administrator to send requests to the RMI Server which in turn talks to the SNMP agent in the appropriate device and obtains the run time values of the requested Object Identifiers (OIDs).

The project also provides features for Set and Get Next operations.



INTRODUCTION

Hereunder, we introduce the entire system along with its system specifications and its advantages.

Functional Specifications:

This section explains the functionality of each part of the MIB Value Reader project.

- ***THE RMI SERVER:***

1. Stores the devices configured and their IP Addresses in a properties file.
2. Stores the MIB file corresponding to each device configured in the server.
3. Stores the xml tree format for each MIB file.
4. Creates and stores configurations created by the client.
5. Stores other information such as administrator password.
6. Talks to a device via its agent.
7. Returns the run-time values, description, syntax and other attributes of the selected OIDs to the client.
8. Maintains a list of the devices which are managed after a single read of the properties file to serve the client.

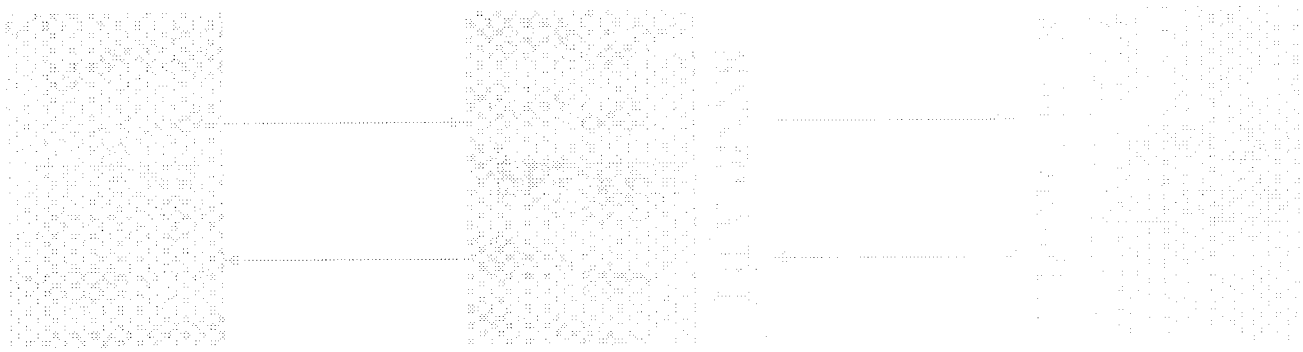
- ***THE CLIENT:***

1. Senses the availability of the server at the specified IP address.
2. Authenticates user on startup and checks if administrator privileges are applicable.
3. Displays the list of devices that are configured as a tree.
4. Displays the values obtained from the server for a single OID at a time.

5. Allows user to create, delete, add OID to, delete OID from, view and evaluate configurations.
6. Displays values of OIDs in a configuration all together.
7. Allows user to add a device to be managed or remove it provided he/she has administrator privileges.

- ***THE DEVICE:***

1. Conforms to the SNMP protocol.
2. Has its MIB data structure.
3. Runs an agent that accepts SNMP requests and reads the MIB data structure and returns values for OIDs.



TECHNOLOGIES / PROTOCOLS USED

Simple Network Management Protocol (SNMP):

The term Simple Network Management Protocol (SNMP) is used to refer to a collection of specifications for network management that includes the protocol itself, the definition of data structures and associated concepts. This protocol is used to monitor and control variables within any host, router, bridge, or other managed device.

SNMP- Related Standards:

The three foundation specifications that define SNMP and its related functions and databases are:

- Structure and identification of management information for TCP/IP- based networks: describes how managed objects contained in the MIB are defined.
- Management Information Base (MIB) for network management of TCP/IP-based internets.
- SNMP defines the protocol used to manage these objects.

NETWORK MANAGEMENT ARCHITECTURE:

The model of network management that is used for TCP/IP network management includes the following key elements:

- Management station
- Managed Device
- Management agent
- Management Information Base
- Network Management Protocol

The management station is typically a stand-alone device, but it may be a capability implemented on a shared system. The management station serves as the interface for the human network manager into the network management system.

The managed device is the one that is managed by the station. The other active element in the network management system is the management agent. Key platforms such as hosts, bridges, routers, and hubs, may be equipped with SNMP agents so that they may be managed from a management station. The management agent responds to requests for information and actions from the management station and may provide the management station with important information.

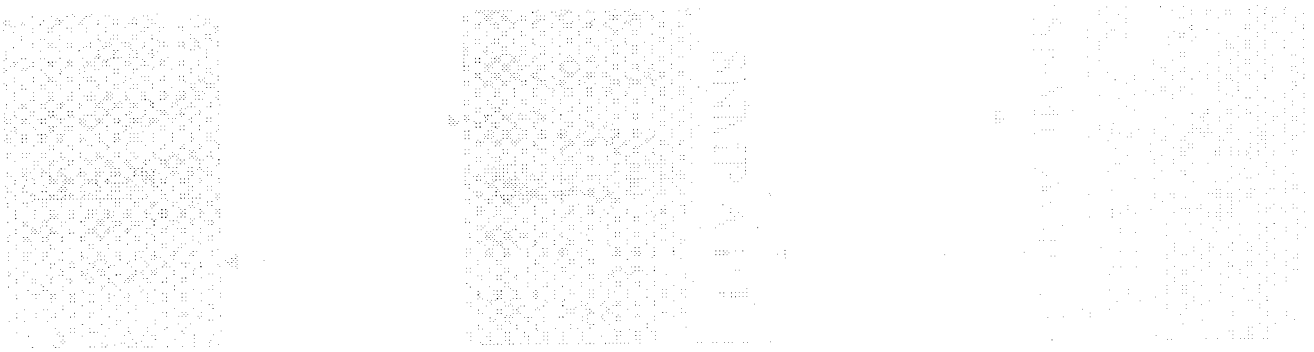
Resources in the network may be managed by representing them as objects. Each object is essentially, a data variable that represents one aspect of the managed agent. The collection of these objects is referred to as a Management Information Base (MIB). A management station performs the monitoring function by retrieving the value of MIB objects.

The management station and agents are linked by a network management protocol. The protocol used for the management of TCP/IP networks is the Simple Network Management Protocol (SNMP), which includes the following key capabilities:

- ***Get:*** enables the management station to retrieve the value of objects at the agent.
- ***Set:*** enables the management station to set the value of objects at the agent.
- ***Get Next:*** enables the management station to retrieve the value of the next object at the agent.
- ***Trap*:** enables an agent to notify the management station of significant events.

Remote Method Invocation (RMI):

Remote Method Invocation is a simple abstraction that makes programming distributed systems in an all-Java environment as simple as writing stand alone Java applications. The main benefit of RMI is ease of rapid prototyping and deployment. RMI uses a registry server for registering and locating objects, but it can then use handles on objects to locate subsequent objects, so accessing remote systems is relatively simple. Marshalling of parameters and return values is taken care of by the system. Focus can be made on application design without too much concern for the communication.



REQUIREMENTS SPECIFICATION DOCUMENT

1. SCOPE

The main purpose of the project is to develop a MIB Value Reader Module using SNMP. It is an application which will connect to any SNMP device and read the MIB values of the device.

2. REFERENCED DOCUMENTS

2.1 Project documents:

- Java 2 Platform API Specifications.

2.2 Third Party / other documents:

- Xerces XML Parser for Java documentation.
- AdventNet SNMP API documentation.

3. REQUIREMENTS

3.1 Functional requirements

- Maintaining the list of devices configured and their IP addresses in the server.
- Returning the list of devices and their corresponding object identifiers as a tree to the client.
- Enabling the user to read run time values of object identifiers of any device configured in the network by talking to the agent of the particular device.
- Enabling the user to add or remove devices provided he/she is an authenticated administrator.
- Allowing the user to create and manage configurations of OIDs and also get values for the OIDs in a configuration.

3.2 Interface requirements

- The server interfaces with the agent via the AdventNet SNMP API.

Get Operation

- Select a device from the device tree
- Construct the corresponding XML tree for the device.
- Select a leaf node from the expanded tree of the device
- Get the attributes of the selected node from the device agent.
- Display the attributes of the leaf node along with the value.

SNMP SET:

Objective / Description	To set a new value for an OID selected by the user at run time.
Primary Users	Network Administrator.
Secondary Users	Network Manager.
Pre-Conditions	The server should be up and responding to requests from the client.

Set Operation

- Select a leaf node from the device tree.
- Enter a new value and its type for the OID represented by the node.

CONFIGURATIONS:

Objective / Description	To enable getting multiple OID values at run time at a single instant.
Primary Users	Network Administrator.
Secondary Users	Network Manager.
Pre-Conditions	The server should be up and responding to requests from the client.

Create Configuration

- Enter the name of configuration file for storing multiple OIDs.
- Store the file in the server.

Add to Configuration

- Select a leaf node from the tree.
- Select the configuration file to which the node is to be added.
- Add the device – OID pair to the configuration file

View Configuration

- Obtain the list of configuration files from the server.
- Select the file from the list of configuration files.
- Obtain the contents of the file.
- Display the contents of the file

Evaluate Configuration

- Select the configuration file
- Send a request for obtaining the values of OIDs present in the file.
- Display the device – OID - value pairs.

Delete Configuration

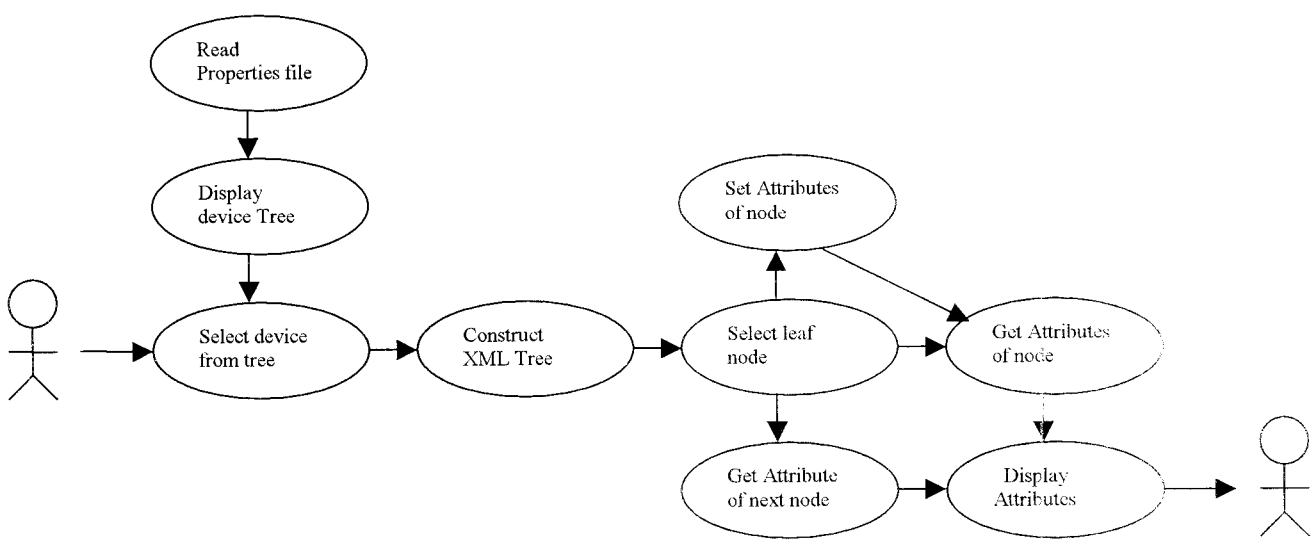
- Select the file to be deleted.
- Delete the file from the server.

Delete OID from Configuration

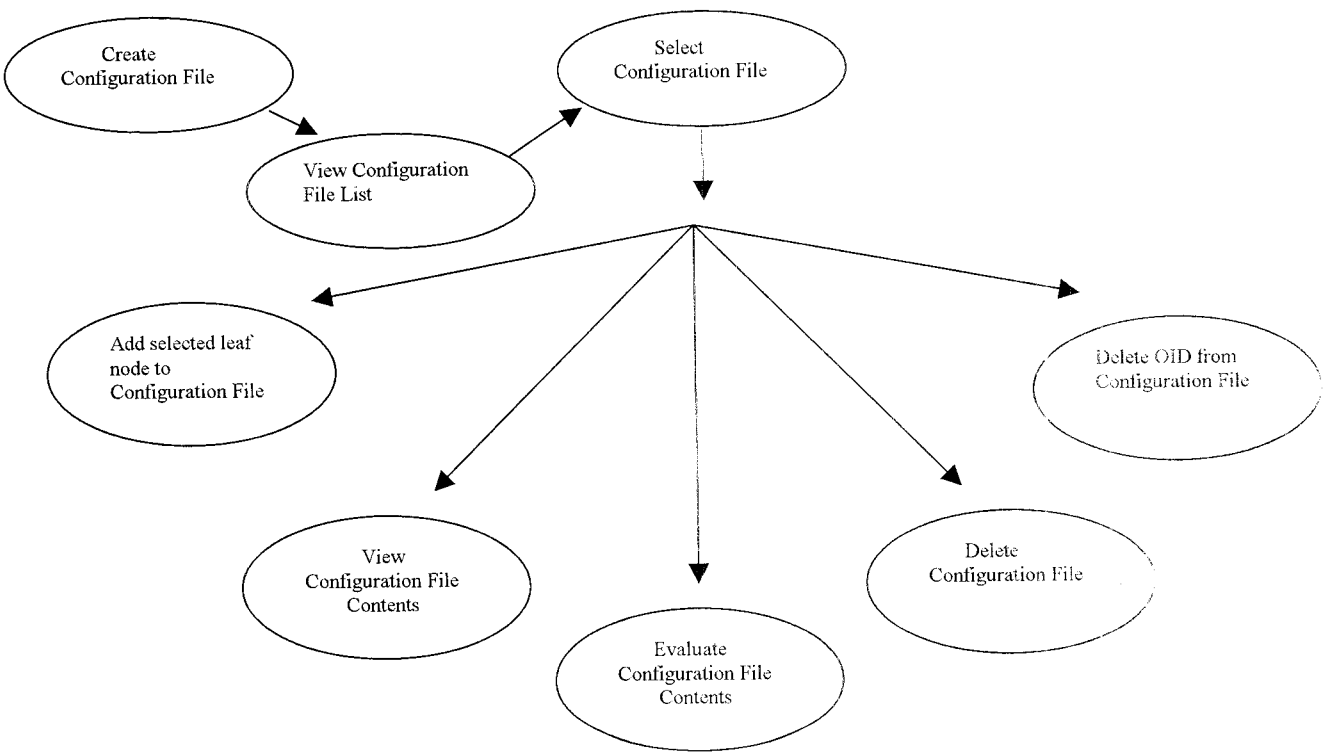
- Select the configuration file
- Select the OID to be deleted
- Delete the device - OID pair from the configuration file in the server.

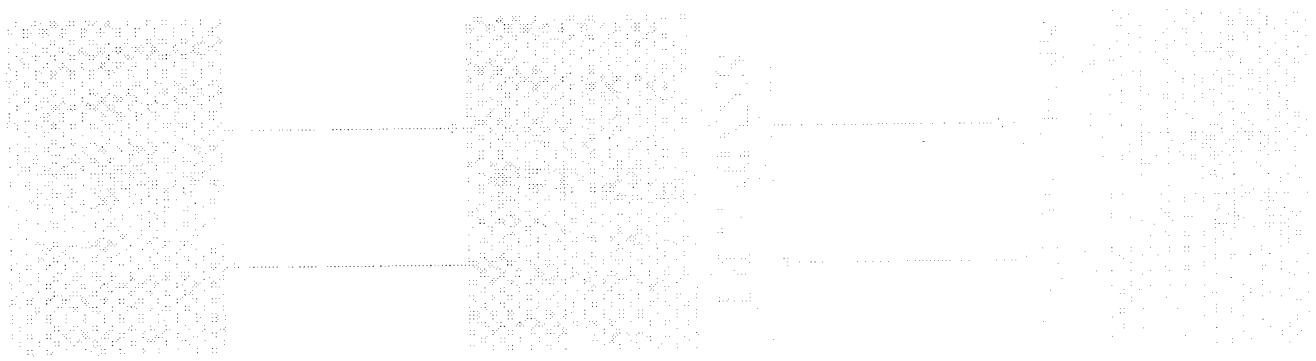
5. USE CASE DIAGRAMS

SNMP OPERATIONS:



CONFIGURATIONS:



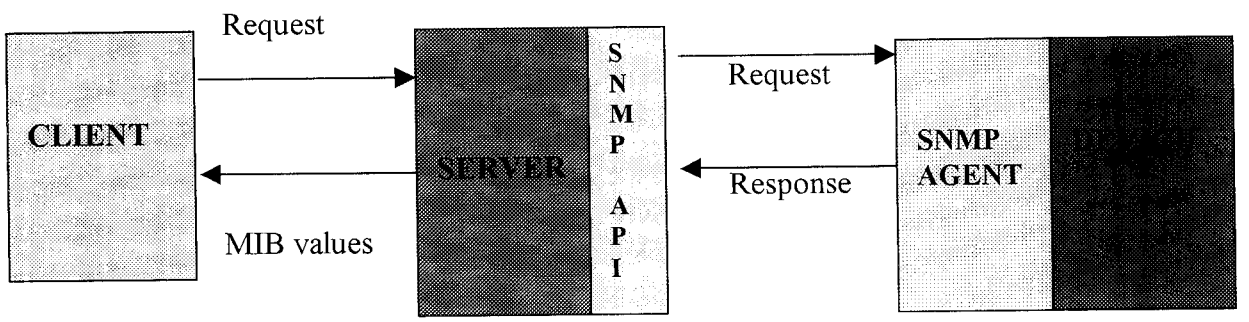


DESIGN DOCUMENT

SYSTEM ARCHITECTURE:

This section explains the basic architecture of the MIBValueReader Server and Client. The architecture explained here is the minimal design that any proposed design should satisfy.

Architecture



The client will connect to the server and the server connects to the device where its SNMP agent runs. The MIBs of the device will be read and values displayed to the client. The application will support any SNMP device. The architecture has the following features.

1. Common client server architecture.
2. The server will be portable to new platforms with minimal effort (means use of Java at the server side).

GUI FLOW:

At client startup, the user is prompted for the administrator password. The user if he/she is the administrator is allowed certain privileges such as adding (configuring) and removing devices that are to be managed. However, a common user may also use the tool by skipping the authentication process without the administrator privileges.

The MIB Value Reader senses the devices that are configured in the server and displays them as a tree. It also provides the following menu options: get, devices, and configuration.

Authentication:

The client provides an input area with a password field, Ok and Cancel buttons. The validity of the entered password is notified with a message box.

Get:

- The user is allowed to select an OID from the tree and obtain its run time value on selection of the get menu item. The syntax, the OID, the value, access and its description are displayed on the right of the tree.

Get Next:

- The user is allowed to select an OID from the tree and obtain the run time value of the next OID on selection of the get-next menu item. The syntax, the OID, the value, access and its description are displayed on the right of the tree.

Set:

- The user is allowed to supply a value along with its type for a selected OID and the value is set in the device's MIB.

Configurations:

The configuration menu provides the following options:

- Create a new configuration.
- View a configuration.
- Add OIDs to a configuration.
- Delete OIDs from a configuration.
- Evaluate a configuration.
- Delete a configuration.

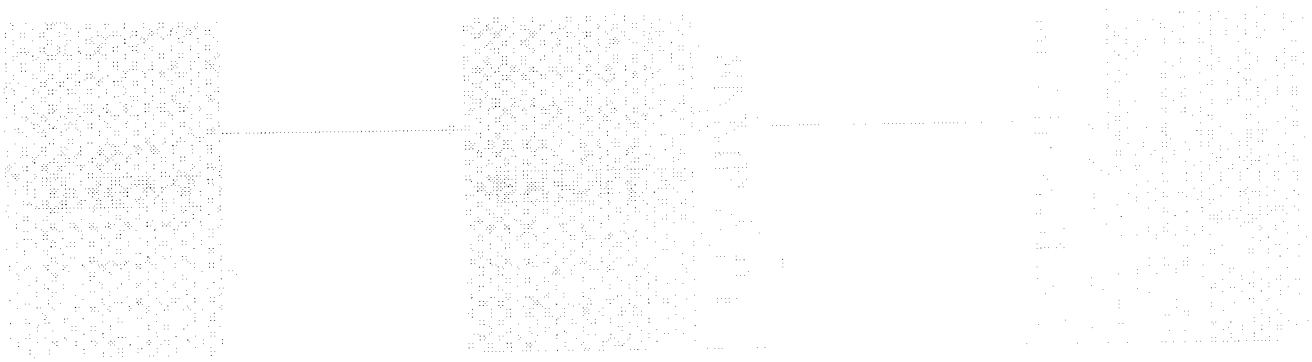
Each of the configuration options provides dialog boxes to take required input from the user and display appropriate messages.

Devices:

This feature allows the user to add new devices and their IP Addresses for monitoring and also remove devices from the device tree when it is no longer needed to be monitored.

The devices menu provides the following options:

- Add device.
- Remove device.



SOURCE CODE

SERVER CODE:

AuthenticationManagerInterface :

```
/*
This interface declares the remote methods for Authentication Manager
class
*/

import java.io.* ;
import java.rmi.server.UnicastRemoteObject;
import java.rmi.*;


public interface AuthenticationManagerInterface extends Remote
{
public boolean isAdministrator(String pass) throws RemoteException;
}
```

AuthenticationManager:

```
/*
This class authenticates the user for access to additional
functionalities like adding, removing devices and Set Operation.
*/

import java.io.* ;
import java.rmi.server.UnicastRemoteObject;
import java.rmi.*;

public class AuthenticationManager extends UnicastRemoteObject
                                implements AuthenticationManagerInterface
{
private static File passFile;
private static FileReader passReader;
private static BufferedReader passBuff;
private static String password, passReceived;

public AuthenticationManager() throws RemoteException
            {
            password = new String();
            }

public boolean isAdministrator(String pass) throws RemoteException
      {
      try{
      passReceived = pass;
      passFile = new File("adminpass.txt");
      passReader = new FileReader(passFile);
```

```
        passBuff = new BufferedReader(passReader);

        password = passBuff.readLine();
        }catch(Exception e){}

        if(passReceived.equals(password))
                {
                return true;
                }
        else
                {
                return false;
                }
        }

}
```

ConfigurationManagerInterface:

```
/*
This interface declares the remote methods for ConfigurationManager
class
*/

import java.io.*;
import java.rmi.* ;
import java.util.*;

public interface ConfigurationManagerInterface extends Remote
        {
        public boolean createConfig(String configurationName) throws
                                                IOException,RemoteException ;
        public boolean addToConfig(String configurationName,
                                String deviceName, String selectedOID)
                                  throws IOException, RemoteException;
        public boolean deleteFromConfig(String
                                configurationFileName,String selectedDevice,
                        String selectedOID) throws FileNotFoundException,
                                                IOException,RemoteException;
        public Vector getConfigDetails(String configurationName) throws
                                               IOException,RemoteException ;
        public boolean deleteConfig(String configurationName) throws
                                                IOException,RemoteException;
        public Vector getConfigFileNames() throws
                                                RemoteException,IOException;


        }
```

ConfigurationManager:

```
/*
 This class provides methods for manipulating configuration files.
*/
```

```
import java.awt.*;
import java.util.*;
import java.io.*;
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;


public class ConfigurationManager extends UnicastRemoteObject
                              implements ConfigurationManagerInterface
      {
      private static String configName,
            configFileName,configStr1,configStr2,configStr,configDevice;
      private static FileOutputStream configFileOutputStream;
      private static String  configOID;
      private static byte buffer[],buffer1[],buffer2[];
      private FileReader configFileReader; //read configuration file
      private File configFile; //config file object
      private BufferedReader configBuffReader;
      private static Vector configVector;
      private static Vector configVector1;

      private static Enumeration configEnum, changeEnum ;
      private static FileInputStream configFileInputStream ;
      private static File configDir;
      private static FilenameFilter filter;
      private static String configArray[];
      int index;
      String temp;



      public ConfigurationManager() throws RemoteException
            {
            super();
            buffer2="\n".getBytes();
            configVector1 = new Vector();
            }

      public boolean createConfig(String configurationName)
                                    throws IOException, RemoteException
            {
            configName = configurationName;
            configFileName = configName + ".config";
            try
            {
            configFileInputStream = new
                                          FileInputStream(configFileName);
            return false;
            }catch(FileNotFoundException e)
            {
            try
            {
            configFileOutputStream = new
                                    FileOutputStream(configFileName,true);

            }catch(FileNotFoundException e1){return false;}
```

```
        }

        configFileOutputStream.close();

        configVector1.addElement(new String(configName));

        return true;

        }

public boolean addToConfig(String configurationName,String
                        selectedDevice, String selectedOID)
                              throws IOException, RemoteException
        {
        configDevice  = selectedDevice;
        configFileName = configurationName+".config";
        configOID = selectedOID;

        try
        {
        configFile = new File(configFileName);
        configFileReader = new FileReader(configFile);
        configBuffReader = new BufferedReader(configFileReader);
        }catch(FileNotFoundException e){return false;}


        while((configStr = configBuffReader.readLine())!=null)
              {
              if(configDevice.equals(configStr))
                    {
                    configStr = configBuffReader.readLine();
                    if(configOID.equals(configStr))
                          {
                          return false;
                          }
                    }
              }

        configBuffReader.close();
        configFileReader.close();

        try
        {
        configFileOutputStream = new
                              FileOutputStream(configFileName,true);
        buffer = configDevice.getBytes();
        buffer1 = configOID.getBytes();

        configFileOutputStream.write(buffer);
        configFileOutputStream.write(buffer2);

        configFileOutputStream.write(buffer1);
        configFileOutputStream.write(buffer2);
        }catch(FileNotFoundException e){return false;}

        configFileOutputStream.close();
```

```
        return true;
        }

//deletes an entry from config file

public boolean deleteFromConfig(String configurationName,String
                        selectedDevice,String selectedOID) throws
                  FileNotFoundException,IOException,RemoteException
{
configFileName = configurationName+".config";
configDevice = selectedDevice;
configOID = selectedOID;
configVector = new Vector();

try
{
configFile = new File(configFileName);
configFileReader = new FileReader(configFile);
configBuffReader = new BufferedReader(configFileReader);
}catch(FileNotFoundException e){return false;}


while((configStr = configBuffReader.readLine())!=null)
        {
        configStr1=configStr;
        if(!configStr1.equals(configDevice))
                {
                configStr = configBuffReader.readLine();
                configStr2=configStr;
                configVector.addElement(new String(configStr1));
                configVector.addElement(new String(configStr2));
                }

        else
                {
                configStr = configBuffReader.readLine();
                configStr2=configStr;
                if(!configStr2.equals(configOID))
                        {
                        configVector.addElement(new
                                                String(configStr1));
                        configVector.addElement(new
                                                String(configStr2));
                        }
                }
        }
        configBuffReader.close();
        configFileReader.close();

try
{
configFileOutputStream = new
                        FileOutputStream(configFileName,false);
}catch(FileNotFoundException e ){return false ;}
```

```
configEnum = configVector.elements();


while(configEnum.hasMoreElements())
      {
      buffer = configEnum.nextElement().toString().getBytes();
      buffer1 = configEnum.nextElement().toString().getBytes();

      configFileOutputStream.write(buffer);
      configFileOutputStream.write(buffer2);
      configFileOutputStream.write(buffer1);
      configFileOutputStream.write(buffer2);

      }
configFileOutputStream.close();
return true ;
}

public Vector getConfigDetails(String configurationName)
                                    throws IOException,RemoteException
{
configFileName = configurationName + ".config";
configVector = new Vector();
configEnum = configVector.elements();
try
{
configFile = new File(configFileName);
configFileReader = new FileReader(configFile);
configBuffReader = new BufferedReader(configFileReader);
}catch(FileNotFoundException e){}


while((configStr = configBuffReader.readLine())!=null)
      {
      configVector.addElement(configStr);
      }

configBuffReader.close();
configFileReader.close();
return configVector;
}

public boolean deleteConfig(String configurationName)
                                    throws IOException,RemoteException
{
try
{
configFileName = configurationName + ".config";
configFile = new File(configFileName);
configFile.delete();


configVector1.remove(configurationName);


}catch(Exception e){return false;}
return true ;
```

```
    }

    public Vector getConfigFileNames()
                                    throws RemoteException,IOException

    {
    System.out.println(configVector1);
    return configVector1;

    }


}
```

DeviceListManagerInterface:

```
//This interface declares remote methods for DeviceListManager Class

import java.io.*;
import java.rmi.* ;
import java.util.*;

public interface DeviceListManagerInterface extends Remote
      {
      public boolean readPropertiesFile()
                                    throws IOException,RemoteException ;
      public Vector getDeviceNames() throws RemoteException ;
      public String getIPAddress() throws RemoteException ;
      public boolean devNameToIPAddress(String devName)
                                    throws RemoteException,IOException;
      public boolean addNewDevice(String newDeviceName,
                String newIPAddress) throws RemoteException,IOException;
      public boolean removeDevice(String devName)
              throws FileNotFoundException,IOException,RemoteException ;

      }
```

DeviceListManager:

```
/*This class provides the methods for access to the properties file
  containing the device names and their ipaddresses.*/

import java.awt.*;
import java.util.*;
import java.io.*;
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;


public class DeviceListManager extends UnicastRemoteObject
                                    implements DeviceListManagerInterface

      {
      private Vector deviceVector;
```

```
private Enumeration venum;
private FileReader devFileReader;
private File devFile;
private FileOutputStream devFileOutputStream;
private BufferedReader devBuffReader;
private BufferedWriter devBuffWriter;
private String devStr;
private boolean devFlag;
private String deviceName;
private String deviceIPAddress;
private byte buffer[],buffer1[],buffer2[];

public DeviceListManager() throws RemoteException
      {
      super();
      buffer2 = "\n".getBytes();
      }

// return status of properties file read.
public boolean readPropertiesFile()
                              throws RemoteException,IOException
      {
      deviceVector = new Vector();
      try
      {
      devFile = new File("devlist.txt");
      devFileReader = new FileReader(devFile);
      devBuffReader = new BufferedReader(devFileReader);
      }catch(FileNotFoundException e){return false;}
      devFlag = true;

      while((devStr = devBuffReader.readLine())!=null)
            {
            if(devFlag)
                  {
                  deviceVector.addElement(new String(devStr));
                  devFlag=false;
                  }
            else
                  devFlag=true;
            }
       devBuffReader.close();
       devFileReader.close();
       return true;
       }

 //returns the vector to the client
 public Vector getDeviceNames() throws RemoteException
       {
       return deviceVector;
       }



 // finds IP address of specified device
 public boolean devNameToIPAddress(String devName)
                                 throws RemoteException,IOException
```

```
        {
        deviceName = devName;
        try
        {
        devFile = new File("devlist.txt");
        devFileReader = new FileReader(devFile);
        devBuffReader = new BufferedReader(devFileReader);
        }catch(FileNotFoundException e){return false;}
        devFlag = true;

        while((devStr = devBuffReader.readLine())!=null)
              {
              if(devFlag)
                    {
                    if(deviceName.equals(devStr))
                          {
                          deviceIPAddress =
                                              devBuffReader.readLine();
                          break;
                          }
                    devFlag=false;
                    }
              else
                    devFlag=true;

              }
         devBuffReader.close();
         devFileReader.close();

         return true;

         }

//returns the IPAddress to the client
public String getIPAddress() throws RemoteException
        {
        return deviceIPAddress;
        }


// adds a new device to the properties file.
public boolean addNewDevice(String newDeviceName,String
                    newIPAddress) throws RemoteException,IOException
        {
        deviceName = newDeviceName;
        deviceIPAddress = newIPAddress ;
        try
        {
        devFile = new File("devlist.txt");
        devFileReader = new FileReader(devFile);
        devBuffReader = new BufferedReader(devFileReader);
        }catch(FileNotFoundException e){return false;}
        devFlag = true;

        while((devStr = devBuffReader.readLine())!=null)
              {
              if(deviceName.equals(devStr))
```

```
                {
                return false;
                }
          else
                {
                devStr = devBuffReader.readLine();
                if(deviceIPAddress.equals(devStr))
                      {
                      return false;
                      }
                }

          }

     devBuffReader.close();
     devFileReader.close();



     devFileOutputStream = new
                           FileOutputStream("devlist.txt",true);
     buffer = deviceName.getBytes();
     buffer1 = deviceIPAddress.getBytes();

     devFileOutputStream.write(buffer);
     devFileOutputStream.write(buffer2);

     devFileOutputStream.write(buffer1);
     devFileOutputStream.write(buffer2);
     devFileOutputStream.close();

     deviceVector.addElement(new String(deviceName));

     return true;
     }


//deletes a device from the properties file
public boolean removeDevice(String devName)
          throws FileNotFoundException,IOException,RemoteException
{
deviceName = devName;

deviceVector = new Vector();

try
{
devFile = new File("devlist.txt");
devFileReader = new FileReader(devFile);
devBuffReader = new BufferedReader(devFileReader);
}catch(FileNotFoundException e){return false;}


while((devStr = devBuffReader.readLine())!=null)
      {
      if(!devStr.equals(deviceName))
            {
```

```
                deviceVector.addElement(new String(devStr));
                devStr = devBuffReader.readLine();
                deviceVector.addElement(new String(devStr));
                }
          else
                {
                devStr = devBuffReader.readLine();
                }
          }
          devBuffReader.close();
          devFileReader.close();

     try
          {
          devFileOutputStream = new
                                FileOutputStream("devlist.txt",false);
          }catch(FileNotFoundException e ){return false ;}

     venum = deviceVector.elements();
     while(venum.hasMoreElements())
          {
          buffer = venum.nextElement().toString().getBytes();
          buffer1 = venum.nextElement().toString().getBytes();

          devFileOutputStream.write(buffer);
          devFileOutputStream.write(buffer2);
          devFileOutputStream.write(buffer1);
          devFileOutputStream.write(buffer2);

          }
     devFileOutputStream.close();
     return true ;
     }
}
```

MIBManagerInterface1:

// This interface declares remote methods for MIBManager Class.

```
import com.adventnet.snmp.mibs.* ;
import java.io.* ;
import java.util.*;
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
import org.w3c.dom.Document ;



public interface MIBManagerInterface1 extends Remote
     {
     //public String getFileURL(String xmlFileName)
                                                throws RemoteException;
     public String getFileContents(String xmlFileName)
                                                throws RemoteException;
     public boolean getMIBFileName(String name)
```

```
                                                    throws RemoteException;

    }
```

MIBManager1:

```
/* This class performs operations such as converting MIB file To XML
format and Obtaining the contents from the XML file.
*/

import com.adventnet.snmp.mibs.* ;
import java.util.*;
import java.rmi.*;
import java.io.* ;
import java.rmi.server.UnicastRemoteObject;
import org.w3c.dom.Document ;
import org.w3c.dom.*;
import org.w3c.dom.NodeList;
import org.w3c.dom.Element ;
import org.apache.xerces.parsers.* ;
import org.apache.xerces.dom.* ;
import java.net.*;


public class MIBManager1 extends UnicastRemoteObject
                                          implements MIBManagerInterface1
    {
    int i;
    private static File devFile;
    private static FileReader devFileReader;
    private static BufferedReader devBuffReader;
    private static String devStr,str;


    private static MibModule mibMode;
    private static String rootNodeName, ext ;
    private static MibNode rootNode ;
    private static Vector v;
    private static Enumeration e ;
    private static byte openTag[];
    private static byte closeTag[];
    private static String openLeftTag= "<" ;
    private static String openRightTag = ">" ;
    private static String closeLeftTag = "</" ;
    private static String closeRightTag = ">" ;
    private static FileOutputStream outFile ;
    private static String deviceName;
    private static String fileName;
    private static Document xmlDocument;
    private static DOMParser domParser;
```

```
URL url;
String urlstr;
File fd;
File fobj;
String str1;
FileWriter fr;
BufferedWriter buff;
String subcont;
File fobj1;
FileWriter fr1;
BufferedWriter buff1;



public MIBManager1() throws RemoteException
      {
      super();
      }

public void MIBToXml(String devName) throws FileNotFoundException
{
deviceName = devName;
fileName = deviceName + ".xml";

MibOperations mibOps = new MibOperations();
try {
    mibMode = mibOps.loadMibModule(deviceName);
    rootNode = mibMode.getRootNode();
    rootNodeName = rootNode.toString();
         }catch (Exception ex) {

                                  }
outFile = new FileOutputStream(fileName);

addOpenTag(rootNodeName);
v = rootNode.getChildList();
e = v.elements();
while(e.hasMoreElements())
       {
       MibNode mibChild = (MibNode)e.nextElement() ;
       String name1 = mibChild.toString();
       addOpenTag(name1);
       if (mibChild.isLeaf())
             {
             addCloseTag(name1);
             }
       else
             {
             recursion(mibChild);
             }
       }
     addCloseTag(rootNodeName);

}
```

```
public static void addOpenTag(String text)
      {
      try{
      openTag = (openLeftTag+ text + openRightTag).getBytes();
      outFile.write(openTag);
      }catch(Exception e){}
      }

public static void addCloseTag(String text)
      {
      try{
      closeTag = (closeLeftTag + text +
                                          closeRightTag).getBytes();
      outFile.write(closeTag);
      }catch(Exception e){}
}

 public static void recursion(MibNode mibnode1)
      {
      Vector v1 = mibnode1.getChildList();
      Enumeration e1 = v1.elements();

      while(e1.hasMoreElements())
      {
      MibNode mibChild2 = (MibNode)e1.nextElement() ;
      String name = mibChild2.toString();
      addOpenTag(name);
      if (mibChild2.isLeaf())
            {
            addCloseTag(name);
            }
      else
            {
            recursion(mibChild2);
            }

      }
      String name3 = mibnode1.toString();
      addCloseTag(name3);
      }


 public String getFileContents(String xmlFileName)
                                          throws RemoteException
      {
      str= new String();
      try{
      devFile = new File(xmlFileName);
      devFileReader = new FileReader(devFile);
      devBuffReader = new BufferedReader(devFileReader);


      while((devStr = devBuffReader.readLine())!=null)
            {
            str = str + devStr + "\n";
            }
```

```
            devBuffReader.close();
            devFileReader.close();

            }catch(Exception e){}
            return str;
            }

      public boolean getMIBFileName(String name) throws RemoteException
            {

            String str = new String();
            str = name;


            try{
            MIBToXml(str);
            }catch(Exception es){}

            return true;
            }

      }
```

SnmpManagerInterface:

```
// This interface declares remote methods for SnmpManager Class.

import com.adventnet.snmp.mibs.* ;
import com.adventnet.snmp.snmp2.*;
import java.io.* ;
import java.util.*;
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;



public interface SnmpManagerInterface extends Remote
      {
      public String getOIDValue(String devName,
            String snmpIPAddr,String snmpOidStr) throws RemoteException ;
      public Vector getConfigValues(String configFileName)
                                    throws RemoteException,IOException;
      public boolean setOIDValue(String devName1, String snmpIPAddr1,
                  String snmpOidStr1,String setValue1,String type1)
                                                throws RemoteException ;

      }
```

SnmpManager:

```
/*
This class performs all the SNMP operations such as Get, Set, Get Next
*/

import java.lang.*;
import java.util.*;
import java.net.*;
import java.rmi.*;
import java.io.*;
import java.rmi.server.UnicastRemoteObject;
import com.adventnet.snmp.snmp2.*;
import com.adventnet.snmp.mibs.* ;

public class SnmpManager extends UnicastRemoteObject
                                        implements SnmpManagerInterface

      {
      private static SnmpSession session ;
      private static SnmpAPI api;
      private static SnmpOID snmpAssignOid,snmpReturnOid,snmpOid ;
      private static SnmpVar snmpVar ;
      private static MibModule mibMode ;
      private static MibOperations mibOps ;
      private static String snmpOIDValueString,snmpDevIPAddress,
              snmpDeviceName,snmpString,snmpMibFileName,snmpIPAddress;
      private static String snmpString1,snmpString2,setValue,setType;
      private static Vector snmpOIDVector,snmpValueVector;
      private static Enumeration snmpOIDEnum,snmpValueEnum;
      private static DeviceListManager snmpDev;
      private static String oidString,snmpAssignOidString;
      private static String devName,snmpIPAddr,snmpOidStr,snmpvarvalue;
      private static MibNode mibNode,mibNextNode;
      private static LeafSyntax leafSyn;
      private static String leafSynStr, description;
      private static String returnString,configName;
      private static String snmpData,deviceName,ipaddress,stringVal;
      private static FileReader file;
      private static BufferedReader snmpBuf;
      private static byte type;
      private static String macroType,accessStr,snmpAssignOidStr;
      private static int access;

      public SnmpManager() throws RemoteException
      {
      super();
      }

      public String getOIDValue(String devName1,
           String snmpIPAddr1,String snmpOidStr1) throws RemoteException
            {
            devName = devName1;
            snmpIPAddr = snmpIPAddr1;
```

```
snmpOidStr = snmpOidStr1;


try{
api = new SnmpAPI();
// add OID
snmpAssignOid = convertStringToOID(devName,snmpOidStr);
snmpAssignOidStr = snmpAssignOid.toString();

leafSyn = mibNode.getSyntax();
leafSynStr = leafSyn.toString();
description = mibNode.getDescription();
access = mibNode.getAccess();
if(access ==(api.NOACCESS))
      accessStr = "Not Accessible";
if(access ==(api.RONLY))
      accessStr = "Read Only";
if(access == (api.RWRITE))
      accessStr = "Read-Write";
if(access ==(api.WONLY))
      accessStr = "Write Only";
if(access ==(api.RCREATE))
      accessStr = "Read-Create";
if(access ==(api.ACCESSFORNOTIFY))
      accessStr = "Accessible-For-Notify";



snmpIPAddress = snmpIPAddr;
// Start SNMP API

api.start();
api.setDebug(false);
}catch(Exception except){System.out.println(except);}

// Open session
session = new SnmpSession(api);
try {
      session.open();
    } catch (SnmpException e )
          {System.err.println("Error opening socket: "+e);}

// set remote host - make sure your agent is running
session.setPeername(snmpIPAddr);


// build GET request
snmpVar = null;
try {
snmpVar = session.get(snmpAssignOid);
} catch (SnmpException e)
{System.err.println("Error sending SNMP request: "+e);}

// print the response pdu

snmpvarvalue = snmpVar.toTagString();
```

```
        // close session
        session.close();
        //stop api thread
        api.close();


        returnString = snmpvarvalue + "==" + leafSynStr + "==" +
           description + "==" + accessStr + "==" + snmpAssignOidStr;
        return returnString;

        }


public SnmpOID convertStringToOID(String snmpDevice,
                                                  String snmpOIDString)

        {
        snmpDeviceName = snmpDevice ;
        snmpString = snmpOIDString ;
        snmpMibFileName = snmpDeviceName + ".mib";
        mibOps = new MibOperations();//changed
        try
        {
        mibMode = mibOps.loadMibModule(snmpDeviceName);
        snmpReturnOid = mibMode.getSnmpOID(snmpString);
        oidString = snmpReturnOid.toString() + ".0";
        snmpReturnOid = new SnmpOID(oidString);
        mibNode = mibMode.getMibNode(snmpReturnOid);
        }
        catch(Exception e){System.out.println(e);}
        return snmpReturnOid;
        }

public Vector getConfigValues(String configFileName)
                              throws RemoteException,IOException

        {
        configName = configFileName + ".config";
        snmpDev = new DeviceListManager();
        snmpValueVector = new Vector();

        file = new FileReader(new File(configName));
        snmpBuf = new BufferedReader(file);
        while((snmpData = snmpBuf.readLine())!=null)
              {
              deviceName = snmpData;
              snmpDev.devNameToIPAddress(snmpData);
              ipaddress = snmpDev.getIPAddress();
              snmpData = snmpBuf.readLine();
              stringVal = getOIDValueLocal(deviceName, ipaddress,
                                                        snmpData);
              snmpValueVector.addElement(new
                   String(deviceName+"::"+snmpData+"::"+stringVal));

              }

        return snmpValueVector;
```

```
    }


public String getOIDValueLocal(String devName1,String
                                    snmpIPAddr1,String snmpOidStr1)
    {
    devName = devName1;
    snmpIPAddr = snmpIPAddr1;
    snmpOidStr = snmpOidStr1;

    try{
    // add OID
    snmpAssignOid = convertStringToOID(devName,snmpOidStr);



    snmpIPAddress = snmpIPAddr;
    // Start SNMP API
    api = new SnmpAPI();
    api.start();
    api.setDebug(false);
    }catch(Exception except){System.out.println(except);}

    // Open session
    session = new SnmpSession(api);
    try {
          session.open();
        } catch (SnmpException e )
          {System.err.println("Error opening socket: "+e);}

    // set remote host - make sure your agent is running
    session.setPeername(snmpIPAddr);



    // build GET request
    snmpVar = null;
    try {
    snmpVar = session.get(snmpAssignOid);
    } catch (SnmpException e)
    {System.err.println("Error sending SNMP request: "+e);}

    // print the response pdu

    snmpvarvalue = snmpVar.toTagString();


    // close session
    session.close();
    //stop api thread
    api.close();


    return snmpvarvalue;

    }
```

```
public boolean setOIDValue(String devName1, String snmpIPAddr1,
        String snmpOidStr1, String setValue1, String type1) throws
                                                        RemoteException

      {
      devName = devName1;
      snmpIPAddr = snmpIPAddr1;
      snmpOidStr = snmpOidStr1;
      setValue = setValue1;
      setType = type1;

      try{
      // add OID
      snmpAssignOid = convertStringToOID(devName,snmpOidStr);

      snmpAssignOidString = snmpAssignOid.toString();
      System.out.println("the oidstring  to be set" +
                                              snmpAssignOidString);


      leafSyn = mibNode.getSyntax();
      leafSynStr = leafSyn.toString();


      snmpIPAddress = snmpIPAddr;
      // Start SNMP API
      api = new SnmpAPI();
      api.start();
      api.setDebug(false);
      }catch(Exception except){System.out.println(except);
                               return false ;}


      if(leafSynStr.equals(setType))
      {
      // Open session
      session = new SnmpSession(api);
      try {
            session.open();
          } catch (SnmpException e )
                 {System.err.println("Error opening socket: "+e);
                 return false ;    }

      // set remote host - make sure your agent is running
      session.setPeername(snmpIPAddr);


      if(setType.equals("DisplayString"))
            { type = api.STRING ;}


      if(setType.equals("INTEGER"))
            {  type = api.INTEGER ;}


      if(setType.equals("TimeTicks"))
            {  type = api.TIMETICKS;}
```

```
if(setType.equals("Counter"))
      {  type = api.COUNTER;}

if(setType.equals("Counter64"))
      {  type = api.COUNTER64;}

if(setType.equals("BITSTRING"))
      {  type = api.BITSTRING;}

if(setType.equals("GAUGE"))
      {  type = api.GAUGE;}


if(setType.equals("IPADDRESS"))
      {  type = api.IPADDRESS;}

if(setType.equals("NETWORKADDRESS"))
      {  type = api.NETWORKADDRESS;}

if(setType.equals("NSAP"))
      {  type = api.NSAP;}

if(setType.equals("NULLOBJ"))
      {  type = api.NULLOBJ;}

if(setType.equals("OBJID"))
      {  type = api.OBJID;}

if(setType.equals("OPAQUE"))
      {  type = api.OPAQUE;}

if(setType.equals("UINTEGER32"))
      {  type = api.UINTEGER32 ;}

if(setType.equals("UNSIGNED32"))
      {  type = api.UNSIGNED32 ;}



// build SET request
snmpVar = null;
try {
snmpVar = session.set(snmpAssignOidString,setValue,type);
    } catch (SnmpException e)
      {System.err.println("Error sending SNMP request: "+e);
                          return false ; }

// print the response pdu

snmpvarvalue = snmpVar.toTagString();


// close session
session.close();
}
else
```

```
        {
        System.err.println("TYPE MISMATCH");
        return false ;
        }

        //stop api thread
        api.close();
        return true ;
        }


public String getNextOIDValue(String devName1, String
            snmpIPAddr1, String snmpOidStr1) throws RemoteException
        {
        devName = devName1;
        snmpIPAddr = snmpIPAddr1;
        snmpOidStr = snmpOidStr1;

        try{
        // add OID
        snmpAssignOid = convertStringToOID(devName,snmpOidStr);


        leafSyn = mibNextNode.getSyntax();
        leafSynStr = leafSyn.toString();
        description = mibNode.getDescription();
        access = mibNode.getAccess();
        if(access ==(api.NOACCESS))
              accessStr = "Not Accessible";
        if(access ==(api.RONLY))
              accessStr = "Read Only";
        if(access == (api.RWRITE))
              accessStr = "Read-Write";
        if(access ==(api.WONLY))
              accessStr = "Write Only";
        if(access ==(api.RCREATE))
              accessStr = "Read-Create";
        if(access ==(api.ACCESSFORNOTIFY))
              accessStr = "Accessible-For-Notify";



        snmpIPAddress = snmpIPAddr;
        // Start SNMP API
        api = new SnmpAPI();
        api.start();
        api.setDebug(false);
        }catch(Exception except){System.out.println(except);}

        // Open session
        session = new SnmpSession(api);
        try {
              session.open();
            } catch (SnmpException e )
                  {System.err.println("Error opening socket: "+e);}
```

```
            // set remote host - make sure your agent is running
            session.setPeername(snmpIPAddr);



            // build GET request
            snmpVar = null;
            try {
            snmpVar = session.getnext(snmpAssignOid);
            } catch (SnmpException e)
            {System.err.println("Error sending SNMP request: "+e);}

            // print the response pdu

            snmpvarvalue = snmpVar.toTagString();


            // close session
            session.close();
            //stop api thread
            api.close();


            returnString = snmpvarvalue + "==" + leafSynStr + "==" +
                  description + "==" + access + "==" +snmpAssignOidStr;
            return returnString;

            }


}
```

StartServer:

```
/*
This class binds all the class objects to the remote registry
*/

import java.rmi.* ;
import java.io.*;
import java.util.*;
import java.rmi.server.UnicastRemoteObject;

public class StartServer
      {
      private static Enumeration enum ;
      private static Vector deviceVector;



      public static void main(String args[])
            {
            System.setSecurityManager(new RMISecurityManager());
            try
                  {
```

```
            AuthenticationManager authenticationManager = new
                                        AuthenticationManager() ;

            Naming.rebind("AuthenticationManager",
                                        authenticationManager);

            DeviceListManager deviceListManager = new
                                         DeviceListManager() ;
            Naming.rebind("DeviceListManager",deviceListManager);

            ConfigurationManager configurationManager = new
                                        ConfigurationManager() ;

            Naming.rebind("ConfigurationManager",
                                         configurationManager);

            MIBManager1 mibManager = new MIBManager1() ;
            Naming.rebind("MIBManager1",mibManager);

            SnmpManager snmpManager = new SnmpManager() ;
            Naming.rebind("SnmpManager",snmpManager);

            System.out.println("server registered");
            }catch(Exception e)
             { System.err.println("error while registering"+ e);}
        }
}
```

CLIENT CODE:

MibValueReaderGUIx:

```
/*
This class provides the methods for creating the GUI of
MibValueReader.
*/


import java.rmi.*;
import java.rmi.registry.*;
import javax.swing.* ;
import javax.swing.tree.*;
import javax.swing.event.*;
import java.awt.* ;
import java.awt.event.*;
import java.io.* ;
import javax.swing.border.*;
import org.w3c.dom.Document ;
import org.w3c.dom.Node;
import org.w3c.dom.*;
import org.w3c.dom.NodeList;
import org.w3c.dom.Element ;
import org.apache.xerces.parsers.* ;
import org.apache.xerces.dom.* ;
import java.util.*;
import org.xml.sax.*;
import java.net.*;
import com.adventnet.snmp.snmp2.*;


public class MibValueReaderGUIx implements WindowListener
      {
      private static JFrame mibFrame;
      private static JMenuBar mibMenuBar;
      private static JMenu mibSnmpMenu,mibDeviceMenu,mibConfigMenu;
      private static JMenuItem getMenuItem, setMenuItem;
      private static JMenuItem addDeviceMenuItem,removeDeviceMenuItem ;
      private static JMenuItem createConfigMenuItem ;
      private static JMenuItem addToConfigMenuItem;
      private static JMenuItem deleteConfigMenuItem;
      private static JMenuItem viewConfigMenuItem ;
      private static JMenuItem evaluateConfigMenuItem;
      private static JSplitPane mibSplitPane;
      private static JScrollPane mibTreeScrollPane;
      private static JScrollPane mibResultAreaScrollPane;
      private static JScrollPane mibSyntaxAreaScrollPane;
```

```
private static JScrollPane mibDescriptionAreaScrollPane;
private static JTextArea  mibDescriptionArea;
private static JTextArea mibResultArea;
private static JTextArea mibSyntaxArea,mibAccessArea,mibOidArea;
private static DeviceListManagerInterface mibDeviceListManager;
private static MIBManagerInterface1 mibMibManager;
private static Vector mibDeviceVector;
private static Enumeration mibEnum;
private static DefaultMutableTreeNode mibDeviceRootNode;
private static DefaultMutableTreeNode mibSubNode;
private static JTree mibDeviceTree;
private static DefaultTreeModel mibDeviceTreeModel;
private static SnmpManagerInterface snmpManager;
private static JLabel syntaxLabel, resultLabel, descLabel;
private JPanel displayPanel;
private static ConfigurationManagerInterface
                                        configurationManager;
private static AuthenticationManagerInterface
                                        authenticationManager;
private static Registry remoteIntr;
private static LocateRegistry locateRegistry;
private static File serverIPAddrFile;
private static FileReader reader;
private static BufferedReader buffReader;
private static JDialog infoDial;
private static JOptionPane opt;


//Main method for the client

public static void main(String args[]) throws
                IOException,RemoteException,NotBoundException
      {
      MibValueReaderGUIx mibValueReader = new
                                        MibValueReaderGUIx();
      mibValueReader.createBasicGUI();

      }



//Constructor that looks up all the registered remote objects

public MibValueReaderGUIx() throws IOException,RemoteException
      {

      try{
      serverIPAddrFile = new File("server ip address.txt");
      reader = new FileReader(serverIPAddrFile);
      buffReader = new BufferedReader(reader);

      remoteIntr =
            locateRegistry.getRegistry(buffReader.readLine());

      mibDeviceListManager =
            (DeviceListManagerInterface)remoteIntr.lookup("Device
```

```
                                                    ListManager");

        mibMibManager =
             (MIBManagerInterface1)remoteIntr.lookup("MIBManager1");

        snmpManager =
             (SnmpManagerInterface)remoteIntr.lookup("SnmpManager");

        configurationManager =
                 (ConfigurationManagerInterface)
                          remoteIntr.lookup("ConfigurationManager");

        authenticationManager =
                 (AuthenticationManagerInterface)
                        Naming.lookup("AuthenticationManager");

        }
        catch(Exception e){opt = new JOptionPane("Server not up and
        ready!",JOptionPane.INFORMATION_MESSAGE,JOptionPane.DEFAULT
        _OPTION);
        infoDial = opt.createDialog(null,"MESSAGE!");
        infoDial.show();
        System.exit(0);
        }

}

//This method authenticates  users.

public void authenticateUser() throws IOException,RemoteException
       {
       JPanel passPanel1 = new JPanel();
       JPanel passPanel2 = new JPanel();
       JLabel passLabel = new JLabel("If you are the
                           administrator, please enter password.");
       JButton ok = new JButton("Ok");
       JButton cancel = new JButton("Cancel");
       JPasswordField passField = new JPasswordField(20);
       passField.setEchoChar('*');
       passField.setPreferredSize(new Dimension(40,40));

       passPanel1.add(passLabel,BorderLayout.NORTH);
       passPanel1.add(passField,BorderLayout.SOUTH);

       FlowLayout flow = new FlowLayout(FlowLayout.CENTER,25,25);
       passPanel2.setLayout(flow);
       passPanel2.add(ok);
       passPanel2.add(cancel);

       JDialog passDialog = new JDialog(mibFrame);
       passDialog.setSize(400,150);

       passDialog.getContentPane().add(passPanel1,
                                              BorderLayout.NORTH);

       passDialog.getContentPane().add(passPanel2,
```

```
                                        BorderLayout.SOUTH);
        System.out.println("Done Display");
        passDialog.setVisible(true);
        passDialog.setLocationRelativeTo(mibFrame);
        ok.addActionListener(new
            AuthenticateOkListener(authenticationManager, passField,
                addDeviceMenuItem, removeDeviceMenuItem, passDialog,
                                                        setMenuItem));
        cancel.addActionListener(new
            AuthenticateCancelListener(passDialog,addDeviceMenuItem,
                                removeDeviceMenuItem, setMenuItem));
        }


//This method creates the Basic GUI

public void createBasicGUI() throws IOException,RemoteException
        {
        //create the frame
        mibFrame = new JFrame();

        mibFrame.getContentPane().setBackground(Color.white);
        mibFrame.setSize(700,550);

        createSplitPanes();
        createMenu();

        mibFrame.getContentPane().add(mibSplitPane,
                                                BorderLayout.CENTER);
        mibFrame.setJMenuBar(mibMenuBar);

        // Display frame.
        mibFrame.setVisible(true);
        authenticateUser();

        }

/*This method creates the tree containing the devices which are
                                                to be managed

*/
public void createDeviceTree() throws IOException,RemoteException
        {

        if(mibDeviceListManager.readPropertiesFile())
                {
                mibDeviceVector =
                                mibDeviceListManager.getDeviceNames();
                }
        mibEnum = mibDeviceVector.elements();



        mibDeviceRootNode=new DefaultMutableTreeNode("Devices");
        while(mibEnum.hasMoreElements())
                {
```

```
            mibSubNode = new
                    DefaultMutableTreeNode(mibEnum.nextElement());
            mibDeviceRootNode.add(mibSubNode);
            }

        mibDeviceTreeModel = new
                    DefaultTreeModel(mibDeviceRootNode);
        mibDeviceTree = new JTree(mibDeviceTreeModel);

        mibDeviceTree.getSelectionModel().setSelectionMode(
                        TreeSelectionModel.SINGLE_TREE_SELECTION);
        }



//This method creates the menu for the application

public void createMenu() throws IOException,RemoteException
        {
        //Create the menu 'n tool bars.
        mibMenuBar = new JMenuBar();

        mibMenuBar.setBorder(BorderFactory.
                                        createRaisedBevelBorder());

        //Build the get menu.
        mibSnmpMenu = new JMenu("SNMP");
        mibSnmpMenu.setMnemonic(KeyEvent.VK_S);
        mibSnmpMenu.getAccessibleContext().
                        setAccessibleDescription("SNMP Options");
        mibMenuBar.add(mibSnmpMenu);


        // Create the SNMP Menu Items.
        getMenuItem = new JMenuItem("Get");
        getMenuItem.addActionListener(new
          GetActionListener(mibDeviceTree,snmpManager,mibResultArea,
                  mibDeviceListManager,mibResultArea, mibSyntaxArea,
                                                mibDescriptionArea));
        mibSnmpMenu.add(getMenuItem);

        setMenuItem = new JMenuItem("Set");
        setMenuItem.addActionListener(new
        SetActionListener(mibDeviceTree,snmpManager,
                                    mibDeviceListManager,mibFrame));
        mibSnmpMenu.add(setMenuItem);


        //build device menu

        mibDeviceMenu = new JMenu("Devices");
        mibDeviceMenu.setMnemonic(KeyEvent.VK_D);
        mibDeviceMenu.getAccessibleContext().
                        setAccessibleDescription("Device Options");
        mibMenuBar.add(mibDeviceMenu);
```

```
// Create the Device Menu Items.
addDeviceMenuItem = new JMenuItem("Add Device");
addDeviceMenuItem.addActionListener(new
      AddDeviceActionListener(mibFrame,mibDeviceListManager,
                  mibDeviceTree,mibMibManager));
mibDeviceMenu.add(addDeviceMenuItem);

removeDeviceMenuItem = new JMenuItem("Remove Device");
removeDeviceMenuItem.addActionListener(new
  RemoveActionListener(mibDeviceTree,mibDeviceListManager));
mibDeviceMenu.add(removeDeviceMenuItem);


//build configuration menu
mibConfigMenu = new JMenu("Configuration");
mibConfigMenu.setMnemonic(KeyEvent.VK_C);
mibConfigMenu.getAccessibleContext().
setAccessibleDescription("Configuration Options");
mibMenuBar.add(mibConfigMenu);

// Create the configuraton Menu Items.
createConfigMenuItem = new JMenuItem("New Configuration");
createConfigMenuItem.addActionListener(new
            CreateConfigActionListener(configurationManager,
                                       mibFrame));
mibConfigMenu.add(createConfigMenuItem);

addToConfigMenuItem = new JMenuItem("Add to
                                          Configuration");
addToConfigMenuItem.addActionListener(new
      AddToConfigActionListener(mibFrame,
            configurationManager,mibDeviceTree));
mibConfigMenu.add(addToConfigMenuItem);


viewConfigMenuItem = new JMenuItem("View Configuration");
viewConfigMenuItem.addActionListener(new
                        ViewConfigActionListener(mibFrame,
                            configurationManager));
mibConfigMenu.add(viewConfigMenuItem);

evaluateConfigMenuItem = new JMenuItem("Evaluate
                                          Configuration");
evaluateConfigMenuItem.addActionListener(new
          EvaluateConfigActionListener(mibFrame,snmpManager,
                            configurationManager));
mibConfigMenu.add(evaluateConfigMenuItem);

deleteConfigMenuItem = new JMenuItem("Delete
                                          Configuration");
deleteConfigMenuItem.addActionListener(new
                       DeleteConfigActionListener(mibFrame,
                            configurationManager));
mibConfigMenu.add(deleteConfigMenuItem);

}
```

```
/*This method divides the frame into two splitpanes and adds the
components to the respective ones.
*/

public void createSplitPanes() throws IOException,RemoteException
      {

      createDeviceTree();
      mibDeviceTree.addTreeSelectionListener(new
       MyTreeSelectionListener(mibDeviceListManager,mibDeviceTree,
                  mibMibManager,getMenuItem,addToConfigMenuItem));

      // create the treescrollpane.
      mibTreeScrollPane = new JScrollPane(mibDeviceTree);



      //Create the resultarea.
      mibResultArea = new JTextArea();
      mibResultArea.setPreferredSize(new Dimension(300,40));
      mibResultArea.setBorder(new LineBorder(Color.lightGray,2));
      mibResultArea.setEditable(true);
      mibResultArea.setBackground(Color.lightGray);
      mibResultArea.setFont(new Font("Times", Font.BOLD, 12));



      //Create the syntaxarea.
      mibSyntaxArea = new JTextArea();
      mibSyntaxArea.setPreferredSize(new Dimension(300,40));
      mibSyntaxArea.setBorder(new LineBorder(Color.lightGray,2));
      mibSyntaxArea.setEditable(true);
      mibSyntaxArea.setBackground(Color.lightGray);
      mibSyntaxArea.setFont(new Font("Times", Font.BOLD, 12));



      //Create the access area.
      mibAccessArea = new JTextArea();
      mibAccessArea.setPreferredSize(new Dimension(300,40));
      mibAccessArea.setBorder(new LineBorder(Color.lightGray,2));
      mibAccessArea.setEditable(true);
      mibAccessArea.setBackground(Color.lightGray);
      mibAccessArea.setFont(new Font("Times", Font.BOLD, 12));



      //Create the oid area.
      mibOidArea = new JTextArea();
      mibOidArea.setPreferredSize(new Dimension(300,30));
      mibOidArea.setBorder(new LineBorder(Color.lightGray,2));
      mibOidArea.setEditable(true);
      mibOidArea.setBackground(Color.lightGray);
      mibOidArea.setFont(new Font("Times", Font.BOLD, 12));
```

```
//Create the description area.
mibDescriptionArea = new JTextArea();
mibDescriptionArea.setPreferredSize(new
                                        Dimension(500,120));
mibDescriptionArea.setBorder(new
                            LineBorder(Color.lightGray,2));
mibDescriptionArea.setEditable(true);
mibDescriptionArea.setBackground(Color.lightGray);
mibDescriptionArea.setFont(new Font("Times",
                                      Font.BOLD,12));



//create the display labels.
syntaxLabel = new JLabel("SYNTAX :          ");
resultLabel = new JLabel("VALUE :           ");
JLabel accessLabel = new JLabel("ACCESS :        ");
JLabel oidLabel = new JLabel("OID :                ");
JLabel descLabel = new JLabel("DESCRIPTION :");


/*Create the textareascrollpane and add the resultsarea
                                                    into it.
*/
int v = ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED;
int h = ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;

mibDescriptionAreaScrollPane = new
                        JScrollPane(mibDescriptionArea,v,h);


FlowLayout flow = new FlowLayout(FlowLayout.LEFT,50,10);
FlowLayout flow1 = new FlowLayout(FlowLayout.LEFT,50,5);


displayPanel = new JPanel();

BoxLayout boxlayout = new
                    BoxLayout(displayPanel,BoxLayout.Y_AXIS);

displayPanel.setLayout(boxlayout);

JPanel pan1 = new JPanel();
JPanel pan2 = new JPanel();
JPanel pan3 = new JPanel();
JPanel pan4 = new JPanel();
JPanel pan5 = new JPanel();

pan1.setLayout(flow);
pan1.add(syntaxLabel);
pan1.add(mibSyntaxArea);

pan2.setLayout(flow);
pan2.add(resultLabel);
```

```
        pan2.add(mibResultArea);

        pan3.setLayout(flow);
        pan3.add(accessLabel);
        pan3.add(mibAccessArea);

        pan4.setLayout(flow);
        pan4.add(oidLabel);
        pan4.add(mibOidArea);

        pan5.setLayout(flow1);
        pan5.setMaximumSize(new Dimension(500,500));
        pan5.add(descLabel);
        pan5.add(mibDescriptionAreaScrollPane);

        displayPanel.add(pan1);
        displayPanel.add(pan2);
        displayPanel.add(pan3);
        displayPanel.add(pan4);
        displayPanel.add(pan5);



        // create splitpanes with the two scrollpanes.
        mibSplitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,
                                      mibTreeScrollPane, displayPanel);
        mibSplitPane.setOneTouchExpandable(false);
        mibSplitPane.setDividerLocation(150);

        //Provide minimum sizes for the two components in the split
                                                                  pane
        mibTreeScrollPane.setMinimumSize(new Dimension(100, 50));
        displayPanel.setMinimumSize(new Dimension(100, 50));

        //Provide a preferred size for the split pane
        mibSplitPane.setPreferredSize(new Dimension(500, 300));
        displayPanel.setPreferredSize(new Dimension(500, 300));

        }

    public void windowClosing(WindowEvent e) {
    mibFrame.setVisible(false); }
    public void windowClosed(WindowEvent e) { System.exit(0); }
    public void windowOpened(WindowEvent e) { }
    public void windowIconified(WindowEvent e) {  }
    public void windowDeiconified(WindowEvent e) { }
    public void windowActivated(WindowEvent e) {  }
    public void windowDeactivated(WindowEvent e) { }

}

//ActionListener for Set Menu Item
class SetActionListener implements ActionListener
{
JDialog setDialog;
JLabel setOIDLabel,setValueLabel,setTypeLabel;
```

```
JTextField setOIDField,setValueField;
JComboBox setTypeCombo;
JButton setOk,setCancel;
JPanel pan1,pan2,pan3,pan4,mainPanel;
String[] typeArray =
{"DisplayString","INTEGER","TimeTicks","Counter","Counter64","BITSTRING
","GAUGE","IPADDRESS","NETWORKADDRESS","NSAP","NULLOBJ","OBJID","OPAQUE
","UINTEGER32","UNSIGNED32"};
JTree mibDevTree;
SnmpManagerInterface snmpMgr;
DeviceListManagerInterface devListMgr;
JFrame mainFrame;

public SetActionListener(JTree mibDeviceTree1,SnmpManagerInterface
snmpManager1,DeviceListManagerInterface mibDeviceListManager1,
JFrame f1)
{
mibDevTree = mibDeviceTree1;
snmpMgr = snmpManager1;
devListMgr = mibDeviceListManager1;
mainFrame = f1;
}

public void actionPerformed(ActionEvent ae)
      {
      setDialog = new JDialog(mainFrame);
      setDialog.setSize(300,300);
      setDialog.setVisible(true);

      setOIDLabel = new JLabel("OID Selected     :");
      setValueLabel = new JLabel("Value to be set  :");
      setTypeLabel = new JLabel("Type              :");

      TreePath path =
                  mibDevTree.getSelectionModel().getSelectionPath();
      Object lastpath = path.getLastPathComponent();

      String oidString = lastpath.toString();

      setOIDField = new JTextField(20);
      setOIDField.setPreferredSize(new Dimension(200,50));
      setOIDField.setEditable(false);
      setOIDField.setText(oidString);

      setValueField = new JTextField(20);
      setValueField.setPreferredSize(new Dimension(200,50));

      setTypeCombo = new JComboBox(typeArray);
      setTypeCombo.setPreferredSize(new Dimension(200,50));

      setOk = new JButton("OK");
      setCancel = new JButton("CANCEL");


      pan1 = new JPanel();
      pan2 = new JPanel();
      pan3 = new JPanel();
```

```
        pan4 = new JPanel();
        mainPanel = new JPanel();

        FlowLayout flow = new FlowLayout(FlowLayout.LEFT,25,25);
        pan1.setLayout(flow);
        pan1.add(setOIDLabel);
        pan1.add(setOIDField);

        pan2.setLayout(flow);
        pan2.add(setValueLabel);
        pan2.add(setValueField);

        pan3.setLayout(flow);
        pan3.add(setTypeLabel);
        pan3.add(setTypeCombo);

        pan4.setLayout(flow);
        pan4.add(setOk);
        pan4.add(setCancel);

        mainPanel.add(pan1);
        mainPanel.add(pan2);
        mainPanel.add(pan3);
        mainPanel.add(pan4);

        setDialog.getContentPane().add(mainPanel);


        setOk.addActionListener(new
                        SetOkActionListener(devListMgr,snmpMgr,mibDevTree,
                                        setDialog,setValueField,setTypeCombo));
        setCancel.addActionListener(new CancelListener(setDialog));

        }
}

//ActionListener for Ok Button of Set Dialog Box
class SetOkActionListener implements ActionListener
{
DeviceListManagerInterface setOkDevMgr;
SnmpManagerInterface setOkSnmpMgr;
JTree setOkTree;
JDialog setOkDialog;
String setOkIPString,setOkDeviceString,setOkValueStr,setOkTypeStr;
JTextField setOkValueField;
JComboBox setOkTypeCombo;
JOptionPane opt;
JDialog infoDial;

public SetOkActionListener(DeviceListManagerInterface
d1,SnmpManagerInterface s1,JTree t1,JDialog d2,JTextField j1,JComboBox
c1)
        {
        setOkValueField = j1;
        setOkTypeCombo = c1;
        setOkDevMgr = d1;
        setOkSnmpMgr = s1;
```

```
        setOkTree = t1;
        setOkDialog = d2;
        }

public void actionPerformed(ActionEvent ae)
        {
        try
        {
        setOkValueStr = setOkValueField.getText();
        setOkTypeStr = (String)setOkTypeCombo.getSelectedItem();


        TreePath path = setOkTree.getSelectionModel().getSelectionPath();
        Object lastpath = path.getLastPathComponent();

        String oidString = lastpath.toString();

        DefaultMutableTreeNode lastselected =
         (DefaultMutableTreeNode)setOkTree.getLastSelectedPathComponent();
        DefaultMutableTreeNode parent =
                        (DefaultMutableTreeNode)lastselected.getParent();

        DefaultMutableTreeNode secondparent = new
                                                DefaultMutableTreeNode();
        while((parent.getUserObject().toString())!="Devices")
              {
              secondparent = (DefaultMutableTreeNode)parent;
              parent = (DefaultMutableTreeNode)parent.getParent();
              }
        setOkDeviceString = secondparent.getUserObject().toString();

        if(setOkDevMgr.devNameToIPAddress(setOkDeviceString))
               {
               setOkIPString = setOkDevMgr.getIPAddress();
               System.out.println("I've got the IP address");
               }

        if(setOkSnmpMgr.setOIDValue(setOkDeviceString,
                        setOkIPString,oidString,setOkValueStr,setOkTypeStr))
              {
              System.out.println("I've set the value");
              opt = new JOptionPane("OID Value
                                    Set.",JOptionPane.INFORMATION_MESSAGE,
                                          JOptionPane.DEFAULT_OPTION);
              infoDial = opt.createDialog(setOkDialog,"MESSAGE!");
              infoDial.show();
              }
        else
              {
              opt = new JOptionPane("Error setting OID
                                  Value.",JOptionPane.INFORMATION_MESSAGE,
                                          JOptionPane.DEFAULT_OPTION);
              infoDial = opt.createDialog(setOkDialog,"MESSAGE!");
              infoDial.show();
              }

        }catch(Exception e){
```

```
            opt = new JOptionPane("OID Value
                                  Set.",JOptionPane.INFORMATION_MESSAGE,
                                        JOptionPane.DEFAULT_OPTION);
            infoDial = opt.createDialog(setOkDialog,"MESSAGE!");
            infoDial.show();}

      setOkDialog.setVisible(false);
      }
}
```

//ActionListener for Ok Button of Authentication dialog box

```
class AuthenticateOkListener implements ActionListener
{
JPasswordField pass1;
String passData;
AuthenticationManagerInterface auth1;
JMenuItem add1, rem1, set1;
JDialog dialog,infoDial;
JOptionPane opt;


public AuthenticateOkListener(AuthenticationManagerInterface auth,
JPasswordField pass, JMenuItem add, JMenuItem rem, JDialog
dial,JMenuItem set)
      {
      dialog = dial;
      add1 = add;
      rem1 = rem;
      auth1 = auth;
      pass1 = pass;
      set1 = set;
      }

public void actionPerformed(ActionEvent ae)
      {
      try{
      passData = pass1.getText();
      if(!auth1.isAdministrator(passData))
            {
            add1.setEnabled(false);
            rem1.setEnabled(false);
            set1.setEnabled(false);
            opt = new JOptionPane("Password
                        incorrect.",JOptionPane.INFORMATION_MESSAGE,
                                    JOptionPane.DEFAULT_OPTION);
            infoDial = opt.createDialog(dialog,"MESSAGE!");
            infoDial.show();
            }
      else
            {
            opt = new JOptionPane("Welcome
                        Administrator.",JOptionPane.INFORMATION_MESSAGE,
                                    JOptionPane.DEFAULT_OPTION);
```

```
            infoDial = opt.createDialog(dialog,"MESSAGE!");
            infoDial.show();
            dialog.setVisible(false);
            }

      }catch(Exception e){}
      }
}
```

//ActionListener for Cancel Button of Authentication dialog Box

```
class AuthenticateCancelListener implements ActionListener
{
JDialog cancelDialog,infoDial;
JMenuItem add1,rem1, set1;
JOptionPane opt;

public AuthenticateCancelListener(JDialog dialogloc,JMenuItem
add,JMenuItem rem, JMenuItem set)
      {
      set1 = set;
      rem1 = rem;
      add1 = add;
      cancelDialog = dialogloc;
      }

public void actionPerformed(ActionEvent ae)
      {
      add1.setEnabled(false);
      rem1.setEnabled(false);
      set1.setEnabled(false);

      opt = new JOptionPane("Not an administrator - some privileges not
                        available.",JOptionPane.INFORMATION_MESSAGE,
                                    JOptionPane.DEFAULT_OPTION);
      infoDial = opt.createDialog(cancelDialog,"MESSAGE!");
      infoDial.show();
      cancelDialog.setVisible(false);
      }

}
```

//Action Listener for Delete Configuration Menu Item

```
class DeleteConfigActionListener implements ActionListener
{
private static JFrame fr;
private static ConfigurationManagerInterface cmgr;
private static Vector cfgList;
private static JList cfgListbox;
private static JDialog dialog, infoDial;
private static JButton addOID,cancel;
private static JPanel configPan,configPan1;
```

```
private static JOptionPane opt;


public DeleteConfigActionListener(JFrame f,
ConfigurationManagerInterface cmi)
      {
      fr =f;
      cmgr = cmi;
      }

public void actionPerformed(ActionEvent ae)
      {
      try
      {
      cfgList = cmgr.getConfigFileNames();
      System.out.println("Srikanth");
      }catch(Exception ce){
            opt = new JOptionPane("No configurations
                        available.",JOptionPane.INFORMATION_MESSAGE,
                                    JOptionPane.DEFAULT_CPTION);
            infoDial = opt.createDialog(fr,"MESSAGE!");
            infoDial.show();
            }
       cfgListbox = new JList(cfgList);

       dialog = new JDialog(fr);
       dialog.setSize(500,500);
       dialog.setVisible(true);

       addOID = new JButton("Delete");
       cancel = new JButton("Cancel");

       configPan = new JPanel();
       configPan1 = new JPanel();
       JScrollPane sc = new JScrollPane(cfgListbox);
       configPan1.add(sc,BorderLayout.CENTER);

       FlowLayout flow = new FlowLayout(FlowLayout.LEFT,15,25);
       configPan.setLayout(flow);

       configPan.add(addOID);
       configPan.add(cancel);

       dialog.getContentPane().add(configPan1,BorderLayout.NORTH);
       dialog.getContentPane().add(configPan,BorderLayout.CENTER);

       addOID.addActionListener(new
           DeleteOkConfigActionListener(cfgListbox,cmgr,dialog,cfgList));
       cancel.addActionListener(new CancelListener(dialog));

       }

}

//ActionListener for Ok Button of Delete Dialog Box

class DeleteOkConfigActionListener implements ActionListener
```

```
{
JList list1;
ConfigurationManagerInterface cm1;
JDialog dial1,infoDial;
Vector vect1;
String selected;
int i;
JOptionPane opt;
boolean status;


public DeleteOkConfigActionListener(JList
        list,ConfigurationManagerInterface cm,JDialog dial,Vector vect)
      {
      list1 = list;
      cm1 = cm;
      dial1 = dial;
      vect1 = vect;
      status = true;
      }

public void actionPerformed(ActionEvent ae)
      {

      try{
      i = list1.getSelectedIndex();
      selected = (String)vect1.elementAt(i);
      }catch(Exception e){opt = new JOptionPane("Select a config to
                              delete.",JOptionPane.INFORMATION_MESSAGE,
                                          JOptionPane.DEFAULT_OPTION);
            infoDial = opt.createDialog(dial1,"MESSAGE!");
            infoDial.show();
            dial1.setVisible(false);
            status = false;
            }


      if(status)
      {
      try
      {
      if(cm1.deleteConfig(selected))
            {
            opt = new JOptionPane("Configuration
                              deleted.",JOptionPane.INFORMATION_MESSAGE,
                                          JOptionPane.DEFAULT_OPTION);
            infoDial = opt.createDialog(dial1,"MESSAGE!");
            infoDial.show();
            }
      }catch(Exception ex){
            opt = new JOptionPane("Select a config to
                              delete.",JOptionPane.INFORMATION_MESSAGE,
                                          JOptionPane.DEFAULT_OPTION);
            infoDial = opt.createDialog(dial1,"MESSAGE!");
            infoDial.show();
            }
```

```
        dial1.setVisible(false);
        }
        }
}


//ActionListener for Obtaining values of OIDs in a configuration

class EvaluateConfigActionListener implements ActionListener
{
JFrame frame;
SnmpManagerInterface snmpManager1;
ConfigurationManagerInterface  configManager1;
Vector cfgList;
JList cfgListbox;
JDialog dialog;
JButton addOID,cancel;
JPanel configPan,configPan1;

public EvaluateConfigActionListener(JFrame f1,SnmpManagerInterface
                                    s1,ConfigurationManagerInterface c1)
        {
        frame = f1;
        snmpManager1 = s1;
        configManager1 = c1;
        cfgList = new Vector();
        }

public void actionPerformed(ActionEvent ae)
        {
        try
        {
        cfgList = configManager1.getConfigFileNames();
        System.out.println("Srikanth");
        }catch(Exception ce){System.out.println("error at getting the
                                                        vector");}

        cfgListbox = new JList(cfgList);

        dialog = new JDialog(frame);
        dialog.setSize(500,500);
        dialog.setVisible(true);

        addOID = new JButton("Obtain Values");
        cancel = new JButton("Cancel");

        configPan = new JPanel();
        configPan1 = new JPanel();
        JScrollPane sc = new JScrollPane(cfgListbox);
        configPan1.add(sc,BorderLayout.CENTER);

        FlowLayout flow = new FlowLayout(FlowLayout.LEFT,15,25);
        configPan.setLayout(flow);


        configPan.add(addOID);
        configPan.add(cancel);
```

```
        dialog.getContentPane().add(configPan1,BorderLayout.NORTH);
        dialog.getContentPane().add(configPan,BorderLayout.CENTER);

        addOID.addActionListener(new
                            ObtainActionListener(cfgListbox,snmpManager1,
                                                 dialog,frame,cfgList));
        cancel.addActionListener(new CancelListener(dialog));

        }
}


/*ActionListener for Obtain Values Button of Evaluate Configuration
                                                          Dialog box

*/
class ObtainActionListener implements ActionListener
{
JList cfgListbox;
SnmpManagerInterface snmpManager2;
JDialog dialog1,dialog,infoDial;
JFrame frame;
Vector cfgList,cfgListNew;
JList configValueResultArea;
JPanel configPan,configPan1;
int i;
String selectedFile;
Enumeration cfgListNewEnum;
JButton add;
JOptionPane opt;

public ObtainActionListener(JList c1,SnmpManagerInterface s1,JDialog
                                                 d1,JFrame f1,Vector v1)
        {
        cfgListbox = c1;
        snmpManager2 = s1;
        dialog1 = d1;
        frame = f1;
        cfgList = v1;
        cfgListNew = new Vector();
        }
 public void actionPerformed(ActionEvent ae)
        {

        if(cfgListbox.isSelectionEmpty())
        {
        opt = new JOptionPane("No config selected to
                            evaluate.",JOptionPane.INFORMATION_MESSAGE,
                                        JOptionPane.DEFAULT_OPTION);
        infoDial = opt.createDialog(dialog1,"MESSAGE!");
        infoDial.show();
        }
        else
        {
        dialog = new JDialog(frame);
        dialog.setSize(500,500);
```

```
    dialog.setVisible(true);


    i = cfgListbox.getSelectedIndex();
    selectedFile = (String)cfgList.elementAt(i);

    try
    {
    cfgListNew = snmpManager2.getConfigValues(selectedFile);

    }catch(Exception es){}


    configValueResultArea = new JList(cfgListNew);
    JScrollPane sc = new JScrollPane(configValueResultArea);
    configPan = new JPanel();
    configPan1 = new JPanel();
    configPan1.add(sc,BorderLayout.CENTER);



    FlowLayout flow = new FlowLayout(FlowLayout.LEFT,15,25);
    configPan.setLayout(flow);


    add = new JButton("Ok");
    configPan.add(add);


    dialog.getContentPane().add(configPan1,BorderLayout.NORTH);
    dialog.getContentPane().add(configPan,BorderLayout.CENTER);

    dialog1.setVisible(false);

    add.addActionListener(new AcceptListener(dialog));

    }//end of else.

    }
}

/*
ActionListener for Ok Button of Dialog Box after obtaining values from
the device
*/

class AcceptListener implements ActionListener
{
JDialog cancelDialog;

public AcceptListener(JDialog dialogloc)
    {
    cancelDialog = dialogloc;
    }

public void actionPerformed(ActionEvent ae)
    {
```

```
        cancelDialog.setVisible(false);
        }

}
//ActionListener for Cancel Button of Evaluate Configuration Dialog Box

class CancelListener implements ActionListener
{
JDialog cancelDialog;

public CancelListener(JDialog dialogloc)
        {
        cancelDialog = dialogloc;
        }

public void actionPerformed(ActionEvent ae)
        {
        cancelDialog.setVisible(false);
        }

}



//Actionlistener to create new configurations

class CreateConfigActionListener implements ActionListener
{
ConfigurationManagerInterface configMan;
JFrame frame;
JDialog dialog;
JTextField configFileName;
JLabel msg;
JButton ok,cancel;
JPanel configPanel1,configPanel2;

public CreateConfigActionListener(ConfigurationManagerInterface
                                                              c,JFrame f

{
configMan = c;
frame = f;
}

public void actionPerformed(ActionEvent ae)
        {
        dialog = new JDialog(frame);
        dialog.setSize(300,300);
        dialog.setVisible(true);

        msg = new JLabel("Enter new Configuration file name");
        configFileName = new JTextField(20);
        ok = new JButton("Ok");
        cancel = new JButton("Cancel");
        configPanel1 = new JPanel();
        configPanel2 = new JPanel();
```

```
        FlowLayout flow = new FlowLayout(FlowLayout.LEFT,25,25);

        configPanel1.add(msg,BorderLayout.CENTER);
        configPanel1.add(configFileName,BorderLayout.SOUTH);


        configPanel2.setLayout(flow);
        configPanel2.add(ok);
        configPanel2.add(cancel);

        dialog.getContentPane().add(configPanel1,BorderLayout.CENTER);
        dialog.getContentPane().add(configPanel2,BorderLayout.SOUTH);

        ok.addActionListener(new
                OkConfigActionListener(dialog,configMan,configFileName));
        cancel.addActionListener(new CancelConfigActionListener(dialog));
        }
}
```

//ActionListener for Ok Button of Create Configuration Dialog box

```
class OkConfigActionListener implements ActionListener
{
JDialog dial1,infoDial;
ConfigurationManagerInterface configMan1;
JTextField text;
String textString;
JOptionPane opt;



public OkConfigActionListener(JDialog d1,ConfigurationManagerInterface
                                                        c1,JTextField t1)
        {
        dial1 = d1;
        configMan1 = c1;
        text = t1;
        }
public void actionPerformed(ActionEvent ae)
        {
        try
        {
        textString = text.getText();

        if(textString.length()==0){
        opt = new JOptionPane("Enter a config
                                name.",JOptionPane.INFORMATION_MESSAGE,
                                        JOptionPane.DEFAULT_OPTION);
        infoDial = opt.createDialog(dial1,"MESSAGE!");
        infoDial.show();
        }

        if(textString.length()!=0)
              {
```

```
            if(configMan1.createConfig(textString))
                  {
                  opt = new JOptionPane("Configuration created. You may
                        add OIDs to it.",JOptionPane.INFORMATION_MESSAGE,
                                          JOptionPane.DEFAULT_OPTION);
                  infoDial = opt.createDialog(dial1,"MESSAGE!");
                  infoDial.show();
                  dial1.setVisible(false);
                  }
            else
                  {
                  opt = new JOptionPane("Configuration already
                              exists.",JOptionPane.INFORMATION_MESSAGE,
                                          JOptionPane.DEFAULT_OPTION);
                  infoDial = opt.createDialog(dial1,"MESSAGE!");
                  infoDial.show();
                  }
            }

      }catch(Exception exec){opt = new JOptionPane("Enter a config
                              name.",JOptionPane.INFORMATION_MESSAGE,
                                          JOptionPane.DEFAULT_OPTION);
            infoDial = opt.createDialog(dial1,"MESSAGE!");
            infoDial.show();}

      }
}

//ActionListener for Cancel Button of Create Configuration Dialog Box

class CancelConfigActionListener implements ActionListener
{
JDialog cancelDialog;

public CancelConfigActionListener(JDialog dialogloc)
      {
      cancelDialog = dialogloc;
      }

public void actionPerformed(ActionEvent ae)
      {
      cancelDialog.setVisible(false);
      }
}


//Action Listener to Add OIDs to the Configuration

class AddToConfigActionListener implements ActionListener
{
JFrame frame2;
ConfigurationManagerInterface configMan3;
JTree deviceTree;
DefaultMutableTreeNode lastselected,parent,secondparent;
String selectedOidString,selectedOidParentString;
String parentOID ;
```

```
JList cfgListbox;
JDialog dialog ;
JButton addOID,cancel;
JPanel configPan;
Vector cfgList;


public AddToConfigActionListener(JFrame
f2,ConfigurationManagerInterface c1,JTree dt)
      {
      frame2 =f2;
      configMan3 = c1;
      deviceTree = dt;
      }
public void actionPerformed(ActionEvent ae)
      {
      cfgList = new Vector();
      lastselected =
      (DefaultMutableTreeNode)deviceTree.getLastSelectedPathComponent();
      selectedOidString = lastselected.toString();
      parent = (DefaultMutableTreeNode)lastselected.getParent();
      parentOID = parent.getUserObject().toString();


      while((parent.getUserObject().toString())!="Devices")
            {
            secondparent = (DefaultMutableTreeNode)parent;
            parent = (DefaultMutableTreeNode)parent.getParent();
            }
      selectedOidParentString = secondparent.getUserObject().
                                                      toString();


      try
      {
      cfgList = configMan3.getConfigFileNames();
      }catch(Exception ce){}
      cfgListbox = new JList(cfgList);

      dialog = new JDialog(frame2);
      dialog.setSize(500,500);
      dialog.setVisible(true);

      addOID = new JButton("Add");
      cancel = new JButton("Cancel");

      JPanel configPan = new JPanel();
      JPanel configPan1 = new JPanel();
      JScrollPane sc = new JScrollPane(cfgListbox);
      configPan1.add(sc,BorderLayout.CENTER);

      FlowLayout flow = new FlowLayout(FlowLayout.LEFT,15,25);
      configPan.setLayout(flow);


      configPan.add(addOID);
      configPan.add(cancel);
```

```
        dialog.getContentPane().add(configPan1,BorderLayout.NORTH);
        dialog.getContentPane().add(configPan,BorderLayout.CENTER);

        addOID.addActionListener(new
           AddOIDActionListener(selectedOidString,selectedOidParentString,
                                cfgListbox,configMan3,dialog,cfgList));
        cancel.addActionListener(new AddCancelListener(dialog));
        }
}
```

//ActionListener for Add Button of AddToConfiguration Dialog Box

```
class AddOIDActionListener implements ActionListener
{
String selectOID;
String parent;
JList list;
ConfigurationManagerInterface config;
JDialog dialog,infoDial;
Vector cfgList;
int index;
String configFileName1;
JOptionPane opt;

public AddOIDActionListener(String s1,String p1,JList
l1,ConfigurationManagerInterface c1,JDialog d1,Vector sr)
       {
       selectOID = s1;
       parent = p1;
       list = l1;
       config = c1;
       dialog = d1;
       cfgList = sr;
       }
 public void actionPerformed(ActionEvent ae)
       {

       if(list.isSelectionEmpty())
       {
       opt = new JOptionPane("Select a config first.",
                                        JOptionPane.INFORMATION_MESSAGE,
                                            JOptionPane.DEFAULT_OPTION);

       infoDial = opt.createDialog(dialog,"MESSAGE!");
       infoDial.show();
       }
       else
       {
       configFileName1 =(String)list.getSelectedValue();
       try
       {
       if(config.addToConfig(configFileName1,parent,selectOID))
             {
             opt = new JOptionPane("OID
                                  added.",JOptionPane.INFORMATION_MESSAGE,
```

```
                                                    JOptionPane.DEFAULT_OPTION);
            infoDial = opt.createDialog(dialog,"MESSAGE!");
            infoDial.show();
            }
      } catch(Exception aec){opt = new JOptionPane("OID not
                                added.",JOptionPane.INFORMATION_MESSAGE,
                                                    JOptionPane.DEFAULT_OPTION);
            infoDial = opt.createDialog(dialog,"MESSAGE!");
            infoDial.show();}

      dialog.setVisible(false);
      }//end of else

      }
}
```

//ActionListener for Cancel Button of AddToConfiguration Dialog Box

```
class AddCancelListener implements ActionListener
{
JDialog cancelDialog;

public AddCancelListener(JDialog dialogloc)
      {
      cancelDialog = dialogloc;
      }

public void actionPerformed(ActionEvent ae)
      {
      cancelDialog.setVisible(false);
      }

}
```

//ActionListener to view the configuration Contents.

```
class ViewConfigActionListener implements ActionListener
{
JFrame frame;
ConfigurationManagerInterface viewConfig;
JList cfgListbox;
JDialog dialog ;
JButton addOID,cancel;
JPanel configPan;
Vector cfgList;


public ViewConfigActionListener(JFrame f1,ConfigurationManagerInterface
c1)
      {
      viewConfig = c1;
      frame = f1;
      cfgList = new Vector();
      }
public void actionPerformed(ActionEvent ae)
```

```
    {
    try
    {
    cfgList = viewConfig.getConfigFileNames();
    }catch(Exception ce){System.out.println("error at getting the
                                                        vector");}

    cfgListbox = new JList(cfgList);

    dialog = new JDialog(frame);
    dialog.setSize(500,500);
    dialog.setVisible(true);

    addOID = new JButton("View");
    cancel = new JButton("Cancel");

    JPanel configPan = new JPanel();
    JPanel configPan1 = new JPanel();
    JScrollPane sc = new JScrollPane(cfgListbox);
    configPan1.add(sc,BorderLayout.CENTER);

    FlowLayout flow = new FlowLayout(FlowLayout.LEFT,15,25);
    configPan.setLayout(flow);


    configPan.add(addOID);
    configPan.add(cancel);

    dialog.getContentPane().add(configPan1,BorderLayout.NORTH);
    dialog.getContentPane().add(configPan,BorderLayout.CENTER);

    addOID.addActionListener(new
                    GetDetailsActionListener(cfgListbox,viewConfig,
                                                dialog,frame,cfgList));
    cancel.addActionListener(new GetDetailsCancelListener(dialog));

    }


}
```

//ActionListener for View Button of View Configuration Dialog Box

```
class GetDetailsActionListener implements ActionListener
{
JList cfgListbox1,cfgListbox;
ConfigurationManagerInterface viewConfig1;
JDialog dialog1;
Vector contentsVector,configViewVector;
int i;
String selectedFile;
JDialog dialog,infoDial ;
JButton addOID,deleteOID;
JPanel configPan;
Vector cfgList1;
```

```
JFrame frame;
Enumeration enum;
String s;
JOptionPane opt;


public GetDetailsActionListener(JList cl,ConfigurationManagerInterface
                                    ml,JDialog dl,JFrame f,Vector vl)

      {
      cfgList1 = vl;
      cfgListbox1 = cl;
      viewConfig1 = ml;
      dialog1 = dl;
      frame = f ;
      configViewVector = new Vector();
      contentsVector = new Vector();

      }

public void actionPerformed(ActionEvent ae)
      {

      if(cfgListbox1.isSelectionEmpty())
            {
            opt = new JOptionPane("No configuration
                        selected.",JOptionPane.INFORMATION_MESSAGE,
                                    JOptionPane.DEFAULT_OPTION);
            infoDial = opt.createDialog(dialog1,"MESSAGE!");
            infoDial.show();
            }
      else
            {

      i = cfgListbox1.getSelectedIndex();
      selectedFile = (String)cfgList1.elementAt(i);
      try
      {
      contentsVector = viewConfig1.getConfigDetails(selectedFile);
      enum = contentsVector.elements();
      while(enum.hasMoreElements())
            {
            s = (String)enum.nextElement() + "::" +
                                    (String)enum.nextElement();
            configViewVector.addElement(new String(s));
            }


      cfgListbox = new JList(configViewVector);
      }catch(Exception e){}

      dialog = new JDialog(frame);
      dialog.setSize(500,500);
      dialog.setVisible(true);

      JLabel descr = new JLabel("Contents of : " + selectedFile);

      addOID = new JButton("Ok");
```

```
        deleteOID = new JButton("Remove OID");

        JPanel configPan = new JPanel();
        JPanel configPan1 = new JPanel();

        JScrollPane sc = new JScrollPane(cfgListbox);

        configPan1.add(descr,BorderLayout.NORTH);
        configPan1.add(sc,BorderLayout.CENTER);

        FlowLayout flow = new FlowLayout(FlowLayout.LEFT,15,25);
        configPan.setLayout(flow);

        configPan.add(addOID);
        configPan.add(deleteOID);


        dialog.getContentPane().add(configPan1,BorderLayout.NORTH);
        dialog.getContentPane().add(configPan,BorderLayout.CENTER);

        addOID.addActionListener(new OkViewActionListener(dialog));
        deleteOID.addActionListener(new
                                DeleteOIDActionListener(viewConfig1,dialog,
                                cfgListbox,configViewVector,selectedFile));

        dialog1.setVisible(false);
        }
        }
}


//ActionListener for deleting OIDs from a selected Configuration

class DeleteOIDActionListener implements ActionListener
{
JDialog dial1;
JList cfgListbox1;
Vector configViewVector1;
String selectedFile1;
ConfigurationManagerInterface config1;
int i;
String listvalue,oid,device;
StringTokenizer tokenizer;
JOptionPane opt;
JDialog infoDial;

public DeleteOIDActionListener(ConfigurationManagerInterface
config,JDialog dial, JList cfgListbox,Vector configViewVector,String
selectedFile)
        {
        dial1 = dial;
        cfgListbox1 = cfgListbox;
        configViewVector1 = configViewVector;
        selectedFile1 = selectedFile;
        config1 = config;
        }
```

```
public void actionPerformed(ActionEvent ae)
      {

      if(cfgListbox1.isSelectionEmpty())
      {
      opt = new JOptionPane("Select OID to
delete.",JOptionPane.INFORMATION_MESSAGE,JOptionPane.DEFAULT_OPTION);
      infoDial = opt.createDialog(dial1,"MESSAGE!");
      infoDial.show();
        }
      else
      {
      try{
      i = cfgListbox1.getSelectedIndex();
      listvalue = (String)configViewVector1.elementAt(i);


      tokenizer = new StringTokenizer(listvalue,"::");
      if(config1.deleteFromConfig(selectedFile1,tokenizer.nextToken()
                                                ,tokenizer.nextToken()))

            {
            opt = new JOptionPane("OID
                        deleted.",JOptionPane.INFORMATION_MESSAGE,
                                    JOptionPane.DEFAULT_OPTION);
            infoDial = opt.createDialog(dial1,"MESSAGE!");
            infoDial.show();
            }
       else
            {
            opt = new JOptionPane("OID not
                        deleted.",JOptionPane.INFORMATION_MESSAGE,
                                    JOptionPane.DEFAULT_OPTION);
            infoDial = opt.createDialog(dial1,"MESSAGE!");
            infoDial.show();
            }
       dial1.setVisible(false);
       }catch(Exception e){}
       }//end of else
       }

  }



  //ActionListener for Cancel Button of View Configuration Dialog Box

  class GetDetailsCancelListener implements ActionListener
  {
  JDialog dial;
  public GetDetailsCancelListener(JDialog d)
        {
        dial = d;
        }
  public void actionPerformed(ActionEvent ae)
        {
        dial.setVisible(false);
```

```
        }
}

//ActionListener for viewing contents of a configuration

class OkViewActionListener implements ActionListener
{
JDialog cancelDialog;

public OkViewActionListener(JDialog dialogloc)
        {
        cancelDialog = dialogloc;
        }

public void actionPerformed(ActionEvent ae)
        {
        cancelDialog.setVisible(false);
        }
}


//ActionListener for performing the Get operation

class GetActionListener implements ActionListener
{
JTree mibDeviceTree1;
SnmpManagerInterface snmpManager1;
JTextArea resultArea1;
DeviceListManagerInterface deviceManager1;
String deviceString,oidString,IPString;
DefaultMutableTreeNode lastselected,parent,secondparent;
String snmpValue;
private static JTextArea  mibDescriptionArea1;
private static JTextArea mibResultArea1, mibSyntaxArea1 ;
private static StringTokenizer tokenizer;
JDialog infoDial;
JOptionPane opt;
Object lastpath;

public GetActionListener(JTree mibDeviceTreeloc, SnmpManagerInterface
snmpManagerloc, JTextArea resultArealoc,DeviceListManagerInterface
deviceManagerloc,JTextArea mibResultArealoc, JTextArea
mibSyntaxArealoc, JTextArea mibDescriptionArealoc)
        {
        deviceManager1 = deviceManagerloc;
        mibDeviceTree1 = mibDeviceTreeloc;
        snmpManager1= snmpManagerloc;
        resultArea1 = new JTextArea();
        resultArea1 = resultArealoc;
        mibResultArea1 = new JTextArea();
        mibResultArea1 = mibResultArealoc;
        mibSyntaxArea1 = new JTextArea();
        mibSyntaxArea1 = mibSyntaxArealoc;
        mibDescriptionArea1 = new JTextArea();
        mibDescriptionArea1 = mibDescriptionArealoc;
        oidString = new String();
```

```
    }
public void actionPerformed(ActionEvent ae)
    {
    try
    {
    TreePath path =
                mibDeviceTree1.getSelectionModel().getSelectionPath();
    lastpath = path.getLastPathComponent();

    oidString = lastpath.toString();

    lastselected = (DefaultMutableTreeNode)mibDeviceTree1
                                        .getLastSelectedPathComponent();
    parent = (DefaultMutableTreeNode)lastselected.getParent();


    while((parent.getUserObject().toString())!="Devices")
          {
          secondparent = (DefaultMutableTreeNode)parent;
          parent = (DefaultMutableTreeNode)parent.getParent();
          }
    deviceString = secondparent.getUserObject().toString();



    if(deviceManager1.devNameToIPAddress(deviceString))
           IPString = deviceManager1.getIPAddress();

    snmpValue =
            snmpManager1.getOIDValue(deviceString,IPString,oidString);

    System.out.println(snmpValue);

    tokenizer = new StringTokenizer(snmpValue,"==");


    mibResultArea1.selectAll();
    mibResultArea1.replaceSelection(tokenizer.nextToken());

    mibSyntaxArea1.selectAll();
    mibSyntaxArea1.replaceSelection(tokenizer.nextToken());


    mibDescriptionArea1.selectAll();
    mibDescriptionArea1.replaceSelection(tokenizer.nextToken());

    }catch(Exception exi)
    {
    opt = new JOptionPane("No OID selected to
                                get!",JOptionPane.INFORMATION_MESSAGE,
                                        JOptionPane.DEFAULT_OPTION);
          infoDial = opt.createDialog(null,"MESSAGE!");
          infoDial.show();}

    }

}
```

```
//ActionListener for adding devices to the Device Tree

class AddDeviceActionListener implements ActionListener
{
JFrame mainFrame1;
DeviceListManagerInterface mibDeviceListManagerpassed1;
JTree tree1;
JDialog dialog;
JPanel pan1,pan2,pan3;
JLabel label1,label2;
JTextField text1,text2;
JButton ok,cancel;
MIBManagerInterface1 mibManager;
JComboBox namesCombo;
String[] deviceNames = { "Win2000", "Win98", "Rfc1158", "Rfc1315",
"Rfc1623", "Rfc1643", "Rfc1289" };

public AddDeviceActionListener(JFrame
mainFrameloc,DeviceListManagerInterface mibDeviceListManagerpassedloc,
JTree treeloc, MIBManagerInterface1 mibloc)
      {
      mainFrame1 = mainFrameloc;
      mibDeviceListManagerpassed1 = mibDeviceListManagerpassedloc;
      tree1 = treeloc;
      mibManager = mibloc;

      }
public void actionPerformed(ActionEvent ae)
      {
      dialog = new JDialog(mainFrame1);
      dialog.setSize(450,400);
      dialog.setVisible(true);

      pan1 = new JPanel();
      pan2 = new JPanel();
      pan3 = new JPanel();

      label1 = new JLabel("Device Name :");
      label2 = new JLabel("IP  Address    :");

      namesCombo = new JComboBox(deviceNames);
      namesCombo.setPreferredSize(new Dimension(100,30));
      text2 = new JTextField(20);
      text2.setPreferredSize(new Dimension(15,25));


      ok=new JButton("Ok");
      cancel=new JButton(" Cancel ");


      FlowLayout flow = new FlowLayout(FlowLayout.LEFT,25,25);
      pan1.setLayout(flow);
      pan1.add(label1);
      pan1.add(namesCombo);
```

```
        pan2.setLayout(flow);
        pan2.add(label2);
        pan2.add(text2);


        FlowLayout flow1 = new FlowLayout(FlowLayout.CENTER,25,25);
        pan3.setLayout(flow1);
        pan3.add(ok);
        pan3.add(cancel);


        dialog.getContentPane().add(pan1,BorderLayout.NORTH);
        dialog.getContentPane().add(pan2,BorderLayout.CENTER);
        dialog.getContentPane().add(pan3,BorderLayout.SOUTH);

        ok.addActionListener(new
                                MIBFileAddActionListener(namesCombo,text2,
                    mibDeviceListManagerpassed1,tree1,dialog,mibManager));
        cancel.addActionListener(new CancelAddActionListener(dialog));
        }
}


//ActionListener for Ok Button of AddDevice Dialog Box

class MIBFileAddActionListener implements ActionListener
{
MIBManagerInterface1 mibMan;
JComboBox oktext1;
JTextField oktext2;
DeviceListManagerInterface okmibDeviceListManagerpassed1;
JTree oktree1;
String text1data,text2data;
DefaultMutableTreeNode mibSubNode,okRoot;
DefaultTreeModel okTreeModel;
JDialog okDialog;
JFileChooser fileChooser;
int returnVal;
File inputFile;
FileReader devFileReader;
BufferedReader devBuffReader;
String devStr, returnStat, str,strInput;

public MIBFileAddActionListener(JComboBox text1loc,JTextField
text2loc,DeviceListManagerInterface
mibDeviceListManagerpassed1loc,JTree tree1loc,JDialog
dialloc,MIBManagerInterface1 mibManloc)
        {
        oktext1 = text1loc;
        oktext2 = text2loc;
        okmibDeviceListManagerpassed1 = mibDeviceListManagerpassed1loc;
        oktree1 = tree1loc;
        okDialog = dialloc;
        mibMan = mibManloc;
```

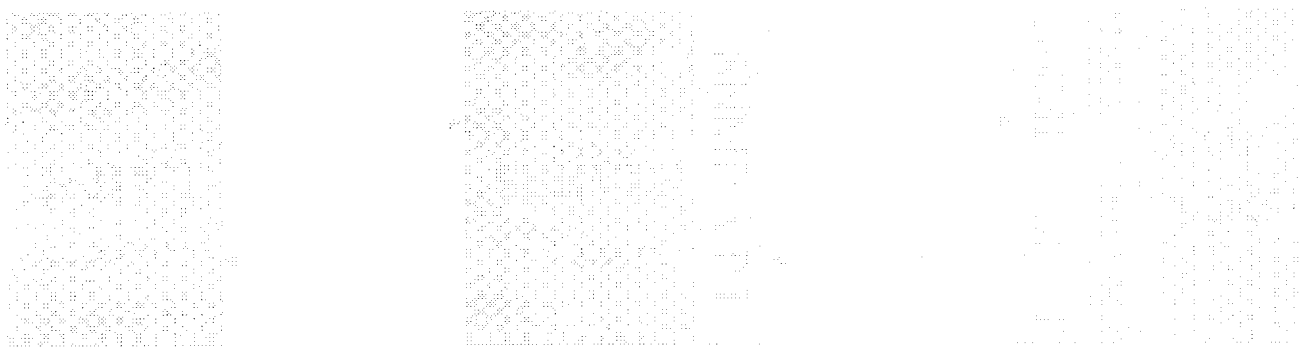
```
    }

public void actionPerformed(ActionEvent ae)
    {
    text1data = (String)oktext1.getSelectedItem();
    text2data = oktext2.getText();
    System.out.println("this ok listener outside try");
    okRoot = (DefaultMutableTreeNode)oktree1.getModel().getRoot();
    okTreeModel = (DefaultTreeModel)oktree1.getModel();
    try
    {
    System.out.println("this ok listener");
    if(okmibDeviceListManagerpassed1.addNewDevice(text1data,
                                                  text2data))
        {
        if(mibMan.getMIBFileName(text1data))
            {
            mibSubNode = new DefaultMutableTreeNode(text1data);
            okTreeModel.insertNodeInto(mibSubNode,okRoot,
                                       okRoot.getChildCount());
            okTreeModel.reload();
            }
        }

    }catch(RemoteException re){}
    catch(IOException ie){}

    okDialog.setVisible(false);

    }

}

//ActionListener for Cancel Button of AddDevice Dialog Box

class CancelAddActionListener implements ActionListener
{
JDialog cancelDialog;

public CancelAddActionListener(JDialog dialogloc)
    {
    cancelDialog = dialogloc;
    }

public void actionPerformed(ActionEvent ae)
    {
    cancelDialog.setVisible(false);
    }
}



//ActionListener for removing devices from the device tree

class RemoveActionListener implements ActionListener
{
```

```
JTree treeRem;
DeviceListManagerInterface mibDeviceListManagerRem;
DefaultMutableTreeNode parentNode,mibSubNode,selNode;
DefaultTreeModel deviceTreeModel;
TreePath path ;
Object lastpath;
String lastpathstr ;

public RemoveActionListener(JTree treeloc,DeviceListManagerInterface
mibDeviceListManagerloc)
      {
      treeRem = treeloc;
      mibDeviceListManagerRem = mibDeviceListManagerloc;
      }
public void actionPerformed(ActionEvent ae)
      {
      deviceTreeModel = (DefaultTreeModel)treeRem.getModel();

      path = treeRem.getSelectionModel().getSelectionPath();
      lastpath = path.getLastPathComponent();
      selNode = (DefaultMutableTreeNode)lastpath;
      parentNode = (DefaultMutableTreeNode)selNode.getParent();

      lastpathstr = lastpath.toString();
      try{
      if(parentNode.getUserObject().toString().equals("Devices"))
            {
            if(mibDeviceListManagerRem.removeDevice(lastpathstr))
                  {
                  mibSubNode =
                  (DefaultMutableTreeNode)(path.getLastPathComponent());
                  System.out.println("node name" + lastpathstr);
                  deviceTreeModel.removeNodeFromParent(mibSubNode);
                  deviceTreeModel.reload();
                  }
            }
       }catch(Exception ex){}


      }

}


//Tree Selection Listener for the device tree.

class MyTreeSelectionListener implements TreeSelectionListener
{
private JTree tree1;
private Vector vect1;
private Enumeration venum1;
private boolean check;
private Object lastpath;
private String lastpathstr, nextel;
private DefaultMutableTreeNode lastselected;
private int lastcount;
```

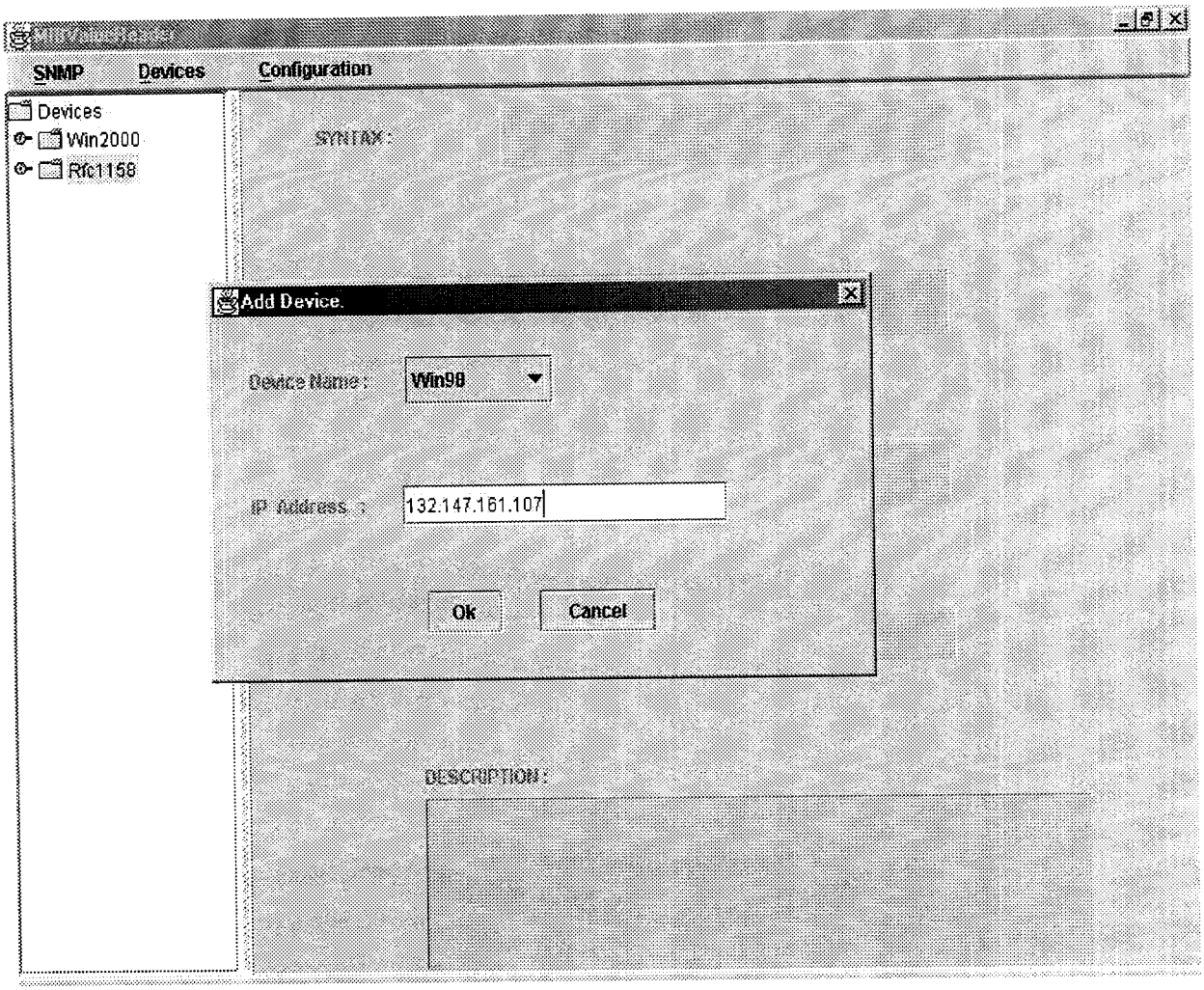
```
private String devname;
private String retfilename;
static MIBManagerInterface1 mibMibManager;
Document xmlDocument ;
File f2,mibFile,removeFile;
Node dsroot;
File dFile;
URL fileurl;
byte buf[];
String fileContents;
private static FileOutputStream fout;
private static DeviceListManagerInterface devMgr;
private static String retFileName;
private static JMenuItem getmain,addTomain;

public MyTreeSelectionListener(DeviceListManagerInterface devMgr1,JTree
tree,MIBManagerInterface1 mib,JMenuItem get,JMenuItem addTo) throws
RemoteException,IOException
      {
      devMgr = devMgr1;
      mibMibManager = mib;
      tree1 = tree;
      check = false ;
      getmain = get;
      addTomain = addTo;
      vect1 = new Vector();
      }


public void valueChanged(TreeSelectionEvent e)
      {
      try{
      if(devMgr.readPropertiesFile())
            {
            vect1 = devMgr.getDeviceNames();
            }
      }catch(Exception except){}

      venum1 = vect1.elements();
      while(venum1.hasMoreElements() && check == false)
            {
            lastselected =
           (DefaultMutableTreeNode)tree1.getLastSelectedPathComponent();

            lastcount = lastselected.getChildCount();

            TreePath path =
                        tree1.getSelectionModel().getSelectionPath();

            if (path != null)
            {
            lastpath = path.getLastPathComponent();
            lastpathstr=lastpath.toString();
            }
            nextel = (String)venum1.nextElement();

            if(lastpathstr.equals(nextel) && lastcount==0)
```

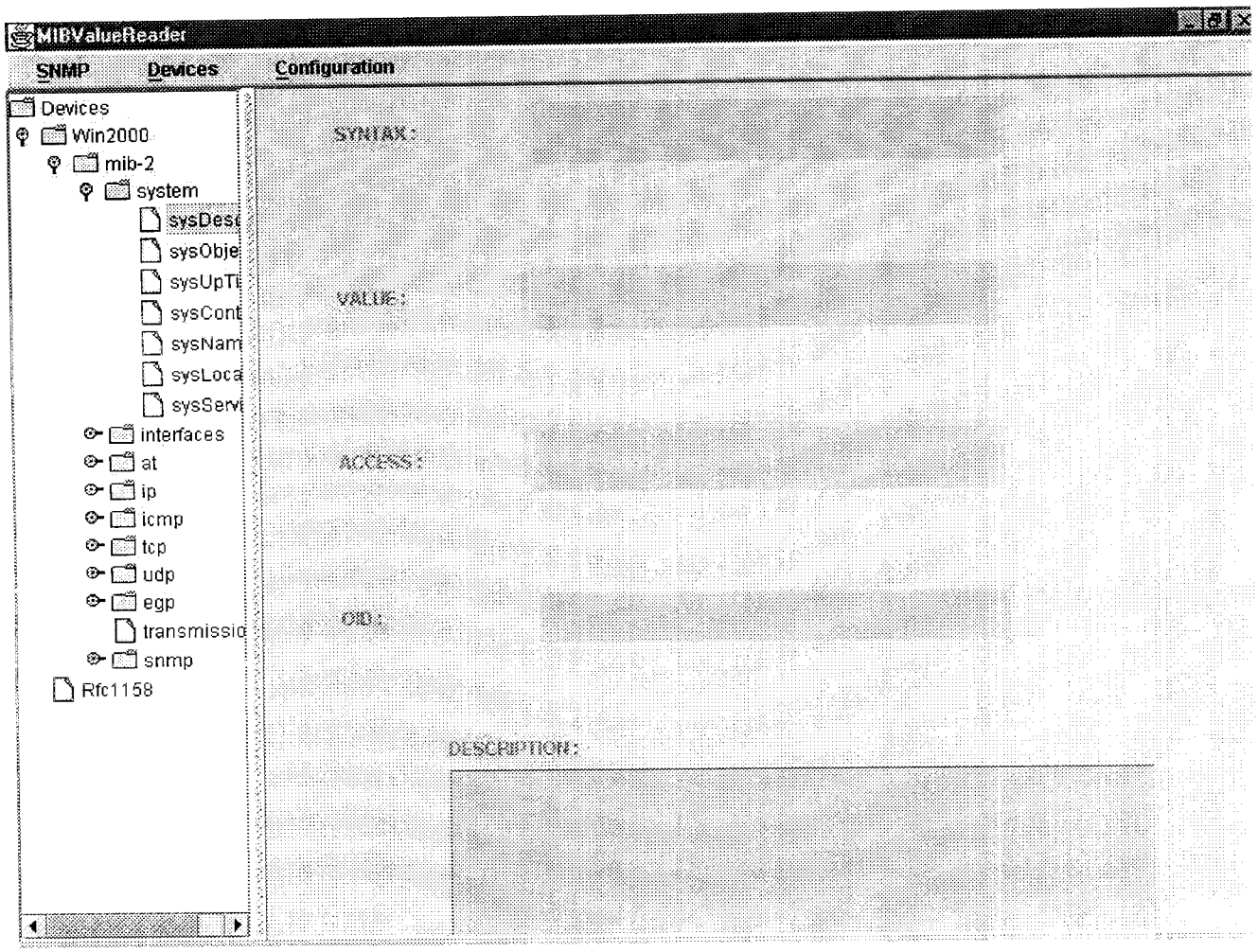
```
        check = true;
        }

if (check == true)
        {
        retFileName = lastselected.toString()+".xml" ;
        System.out.println(lastselected.toString());

        try{
        fileContents = mibMibManager.getFileContents(retFileName);

        fout = new FileOutputStream(retFileName,true);
        fout.write(fileContents.getBytes());
        fout.close();

        }catch(Exception excep){System.out.println("sdfsadfadf");}

        try
        {
        DOMParser parser = new DOMParser();
        parser.parse(retFileName);
        Document doc = parser.getDocument();
        Node dsroot = doc.getDocumentElement();
        System.out.println(dsroot.getNodeName());



        for(Node dschild =
              dsroot.getFirstChild();dschild!=null;
                                    dschild =dschild.getNextSibling()
              {

              DefaultMutableTreeNode node = new
                      DefaultMutableTreeNode(dschild.getNodeName());
              DefaultMutableTreeNode n3 =
                                       (DefaultMutableTreeNode)(tree1.
                                     getLastSelectedPathComponent());
              n3.add(node);
              if(dschild.hasChildNodes())
              buildChildren(dschild, node);
              }

        check = false;
        }catch(IOException ex){}
         catch(SAXException sax){}
        }

 removeFile = new File(retFileName);
 removeFile.delete();

 }
```

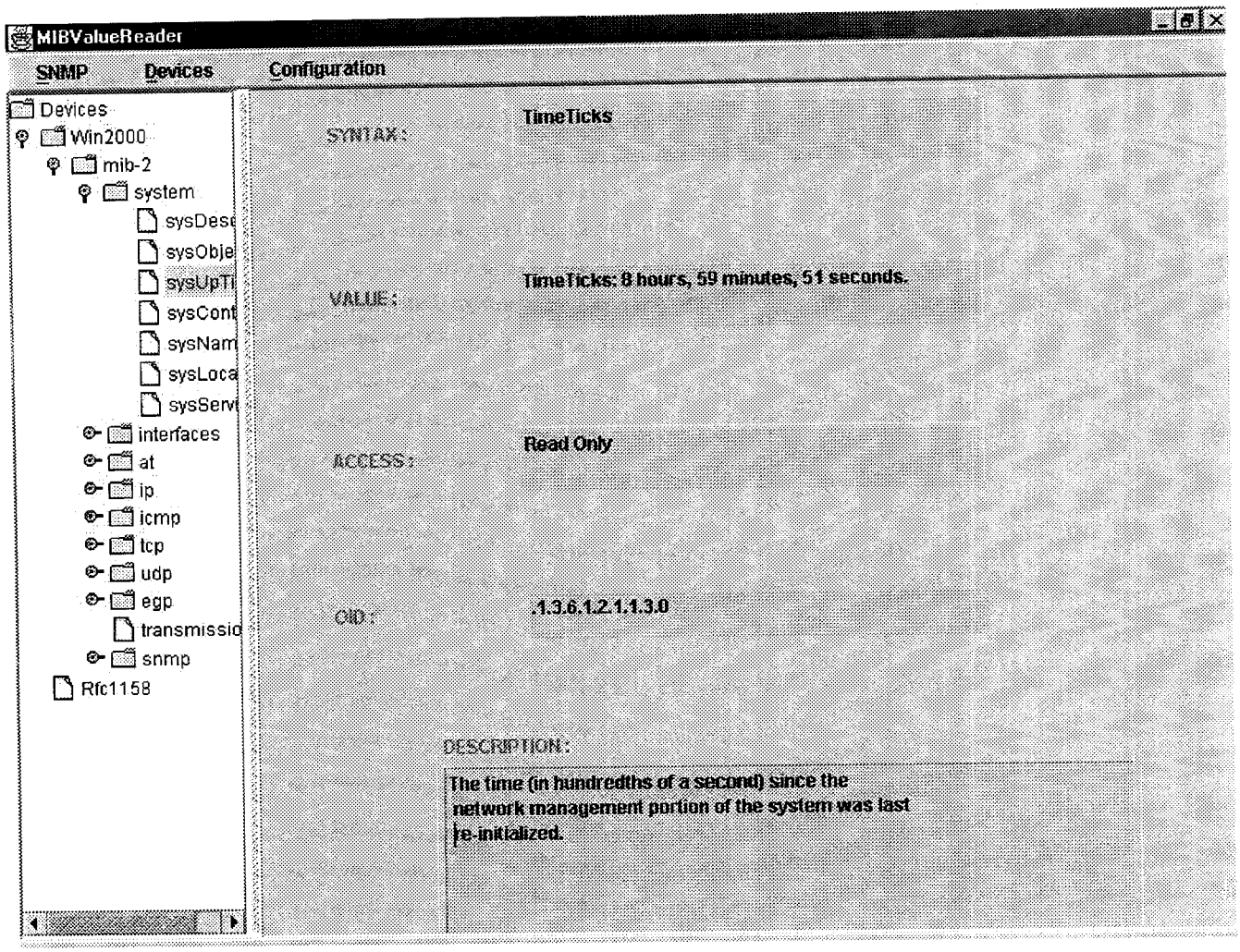
```
public void buildChildren(Node child,DefaultMutableTreeNode node)
      {
      for(Node n = child.getFirstChild();n!=null;
                                    n = n.getNextSibling())
            {
            DefaultMutableTreeNode node1 = new
                        DefaultMutableTreeNode(n.getNodeName());
            node.add(node1);
            if(child.hasChildNodes())
            buildChildren(n,node1);
            }
      }

}
```

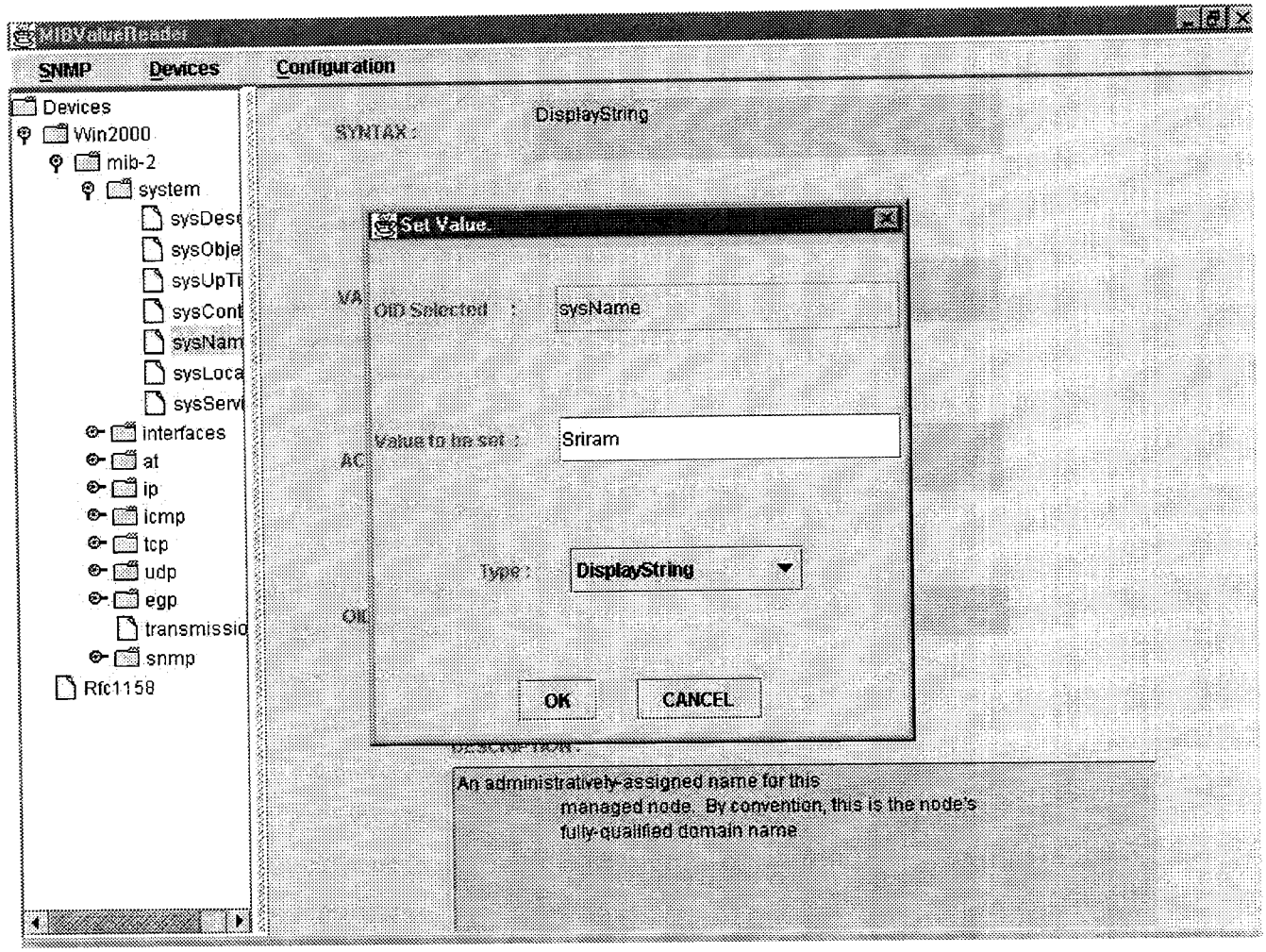


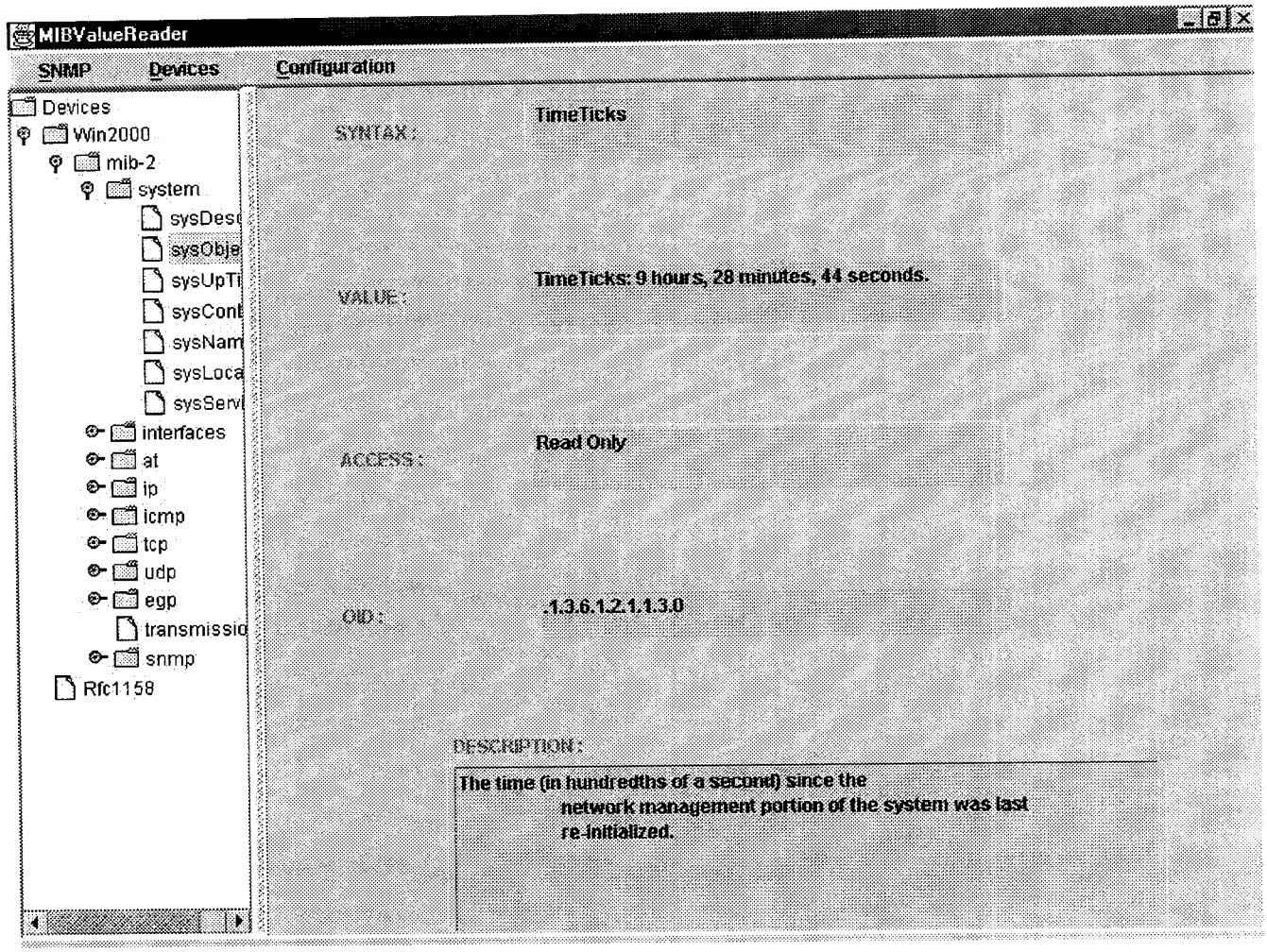
SCREEN SNAPSHOTS

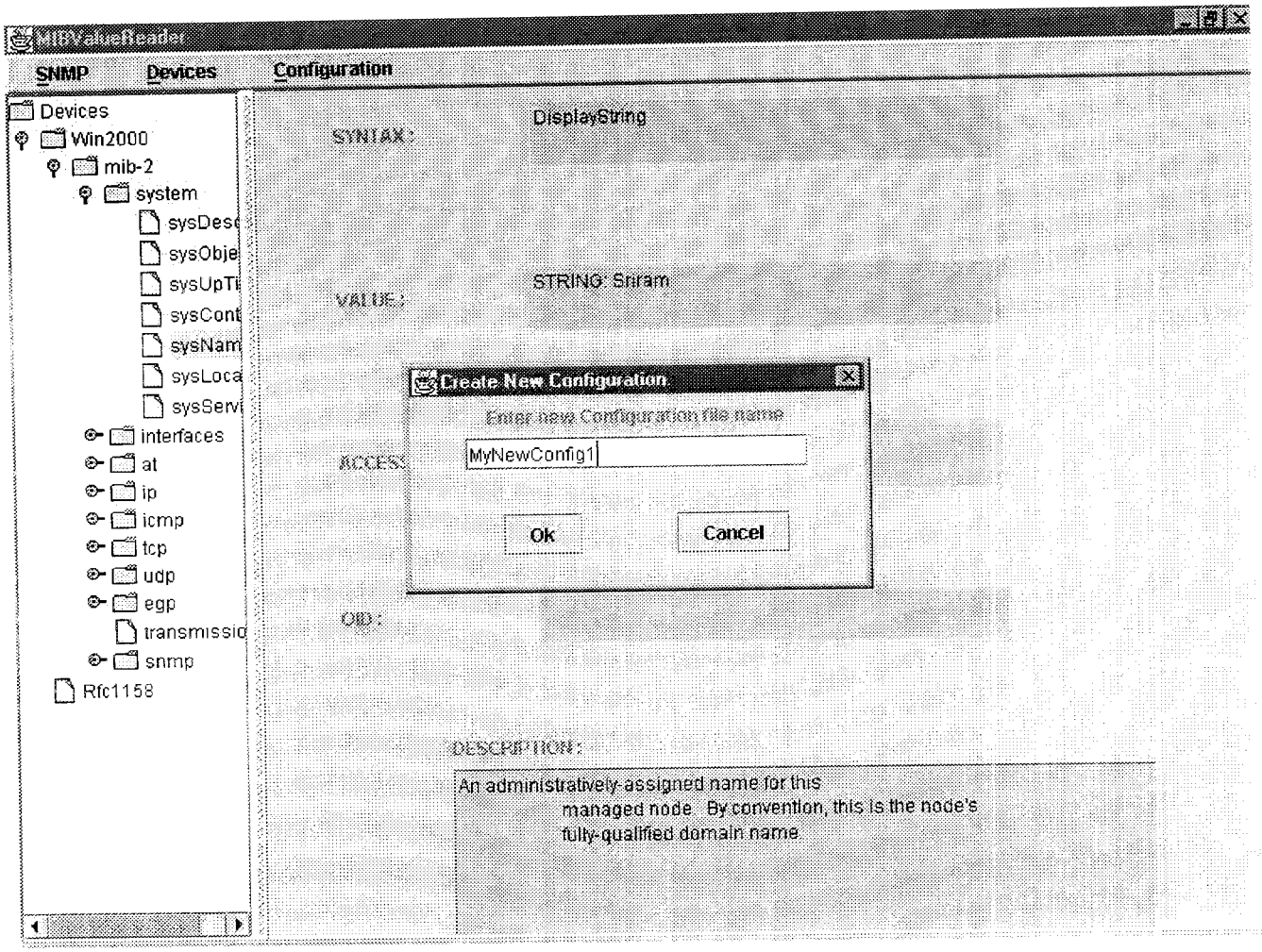

SNMP
Devices
Configuration
Devices
Win2000
Rfc1158
SYNTAX :
Add Device.
Device Name :
Win98
IP Address :
132.147.161.107
Ok
Cancel
DESCRIPTION :

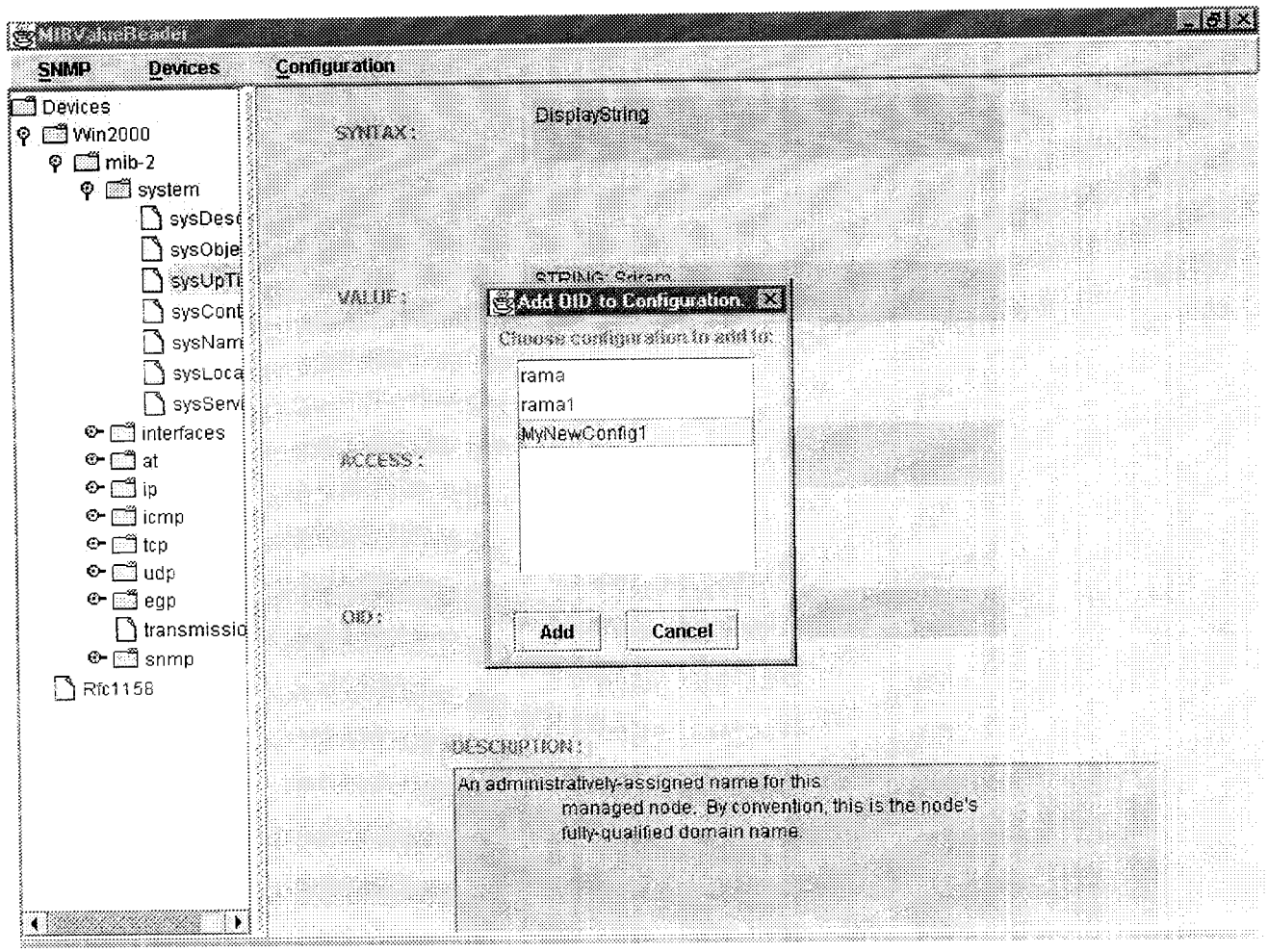

MIBValueReader
SNMP
Devices
Configuration
Devices
Win2000
mib-2
system
sysDesc
sysObje
sysUpTi
sysCont
sysNam
sysLoca
sysServ
interfaces
at
ip
icmp
tcp
udp
egp
transmissio
snmp
Rfc1158
SYNTAX:
VALUE:
ACCESS:
OID:
DESCRIPTION:

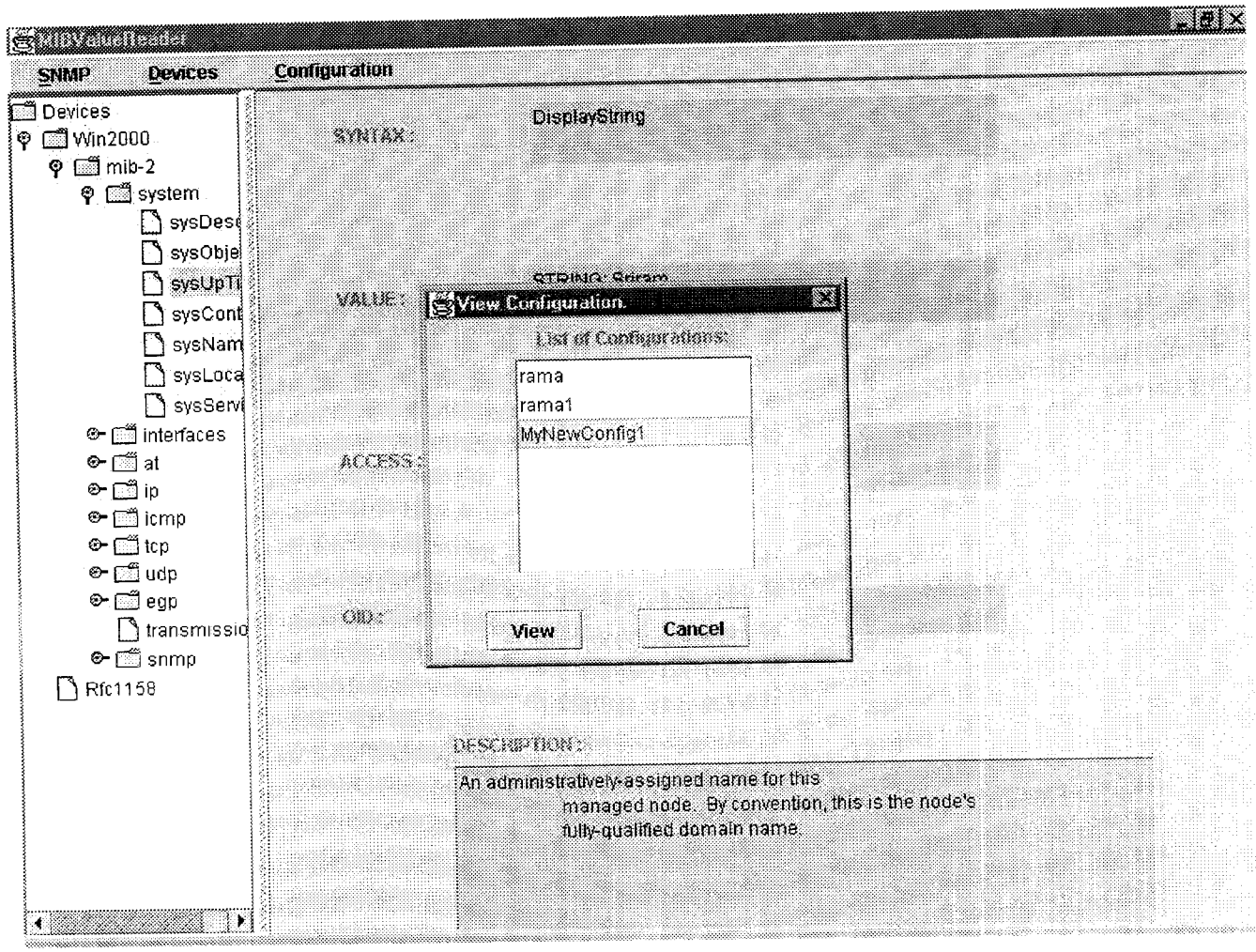

MIBValueReader
SNMP
Devices
Configuration
Devices
Win2000
mib-2
system
sysDesc
sysObje
sysUpTi
sysCont
sysNam
sysLoca
sysServ
interfaces
at
ip
icmp
tcp
udp
egp
transmissio
snmp
Rfc1158
SYNTAX:
TimeTicks
VALUE:
TimeTicks: 8 hours, 59 minutes, 51 seconds.
ACCESS:
Read Only
OID:
.1.3.6.1.2.1.1.3.0
DESCRIPTION:
The time (in hundredths of a second) since the network management portion of the system was last re-initialized.

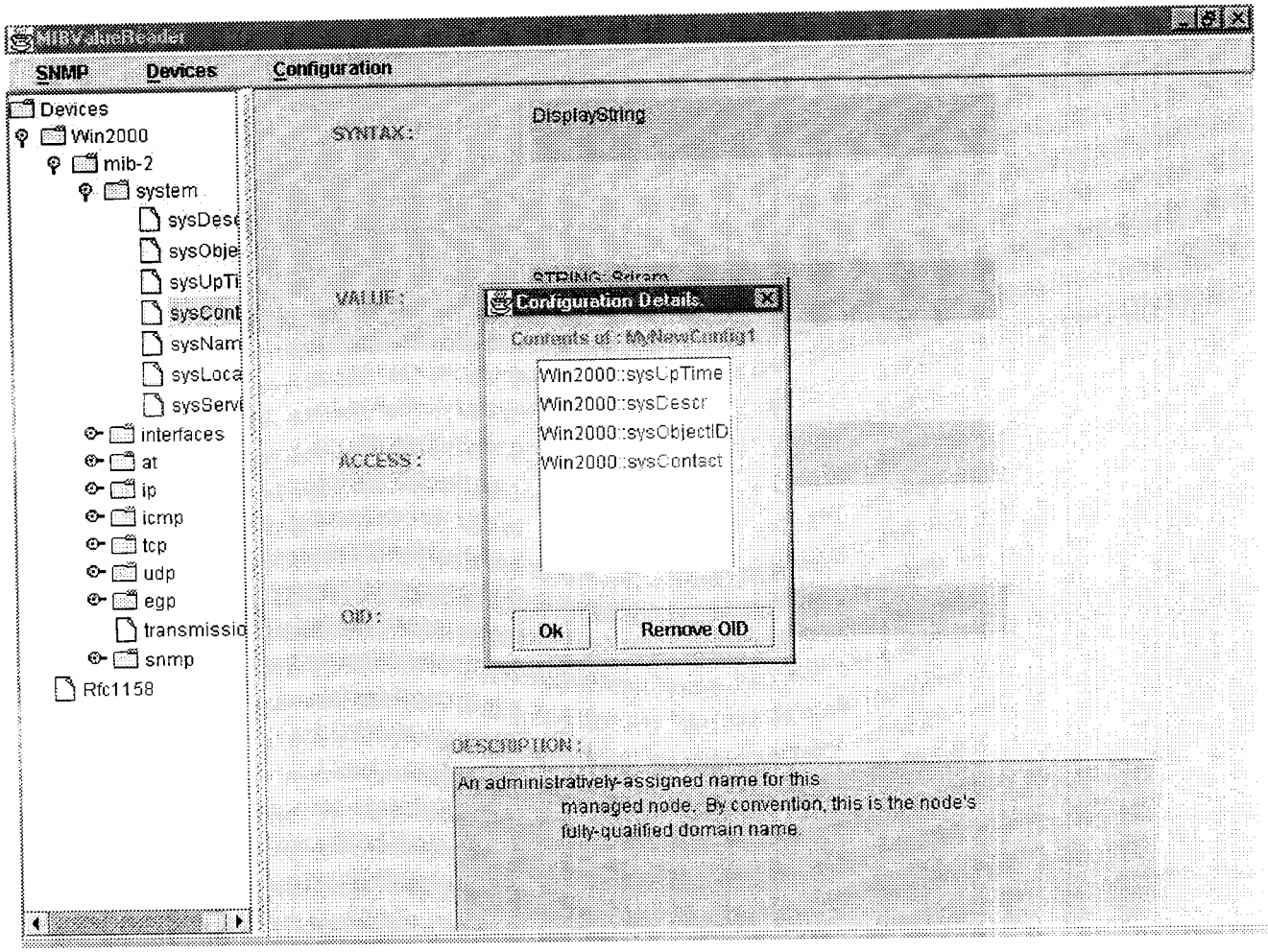

MIBValueReader
SNMP
Devices
Configuration
Devices
Win2000
mib-2
system
sysDesc
sysObje
sysUpTi
sysCont
sysNam
sysLoca
sysServ
interfaces
at
ip
icmp
tcp
udp
egp
transmissio
snmp
Rfc1158
SYNTAX :
DisplayString
VA
AC
OB
Set Value.
OID Selected :
sysName
Value to be set :
Sriram
Type :
DisplayString
OK
CANCEL
An administratively-assigned name for this
managed node. By convention, this is the node's
fully-qualified domain name

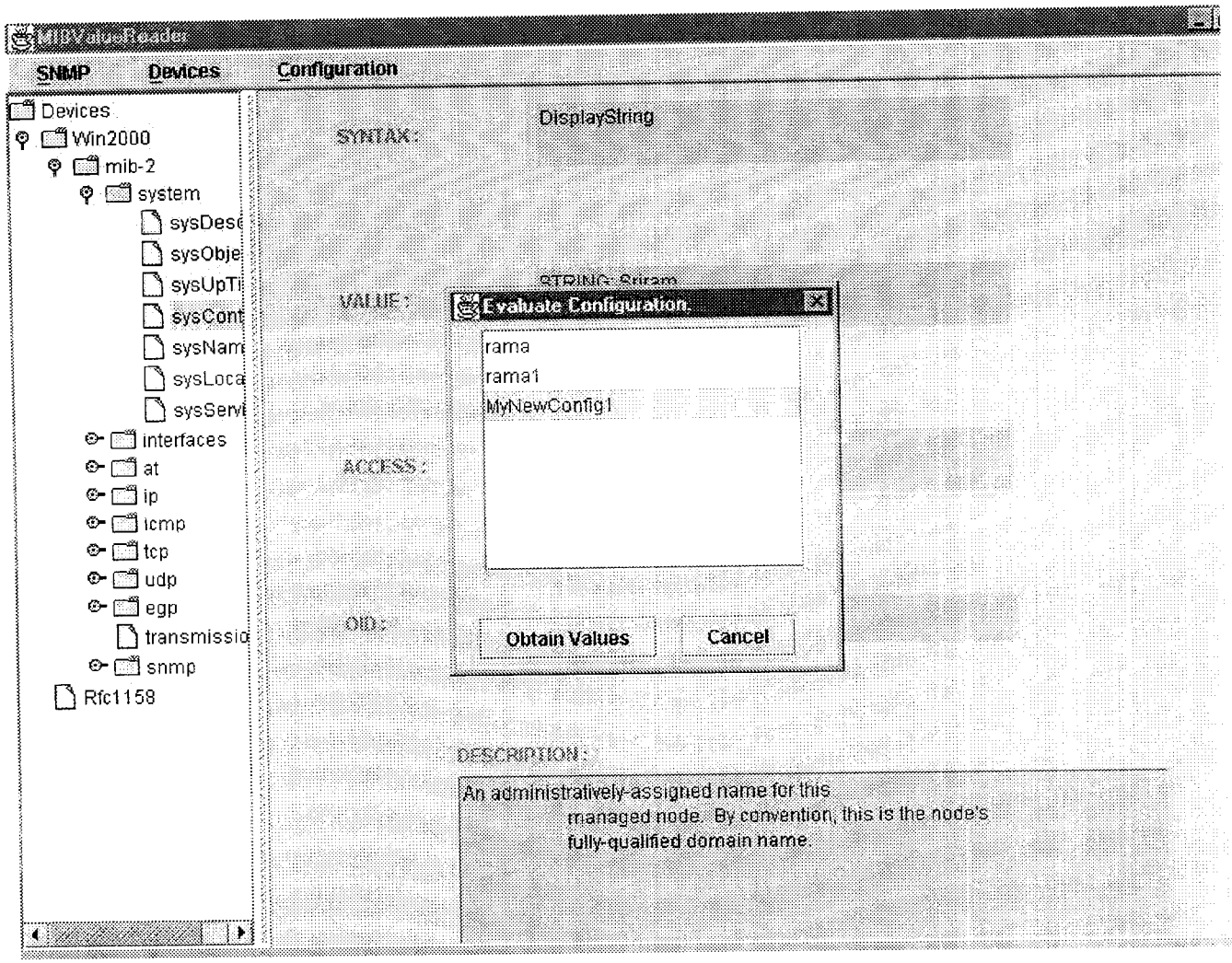

MIBValueReader
SNMP
Devices
Configuration
Devices
Win2000
mib-2
system
sysDesc
sysObje
sysUpTi
sysCont
sysNam
sysLoca
sysServ
interfaces
at
ip
icmp
tcp
udp
egp
transmissio
snmp
Rfc1158
SYNTAX:
TimeTicks
VALUE:
TimeTicks: 9 hours, 28 minutes, 44 seconds.
ACCESS:
Read Only
OID:
.1.3.6.1.2.1.1.3.0
DESCRIPTION:
The time (in hundredths of a second) since the
network management portion of the system was last
re-initialized.

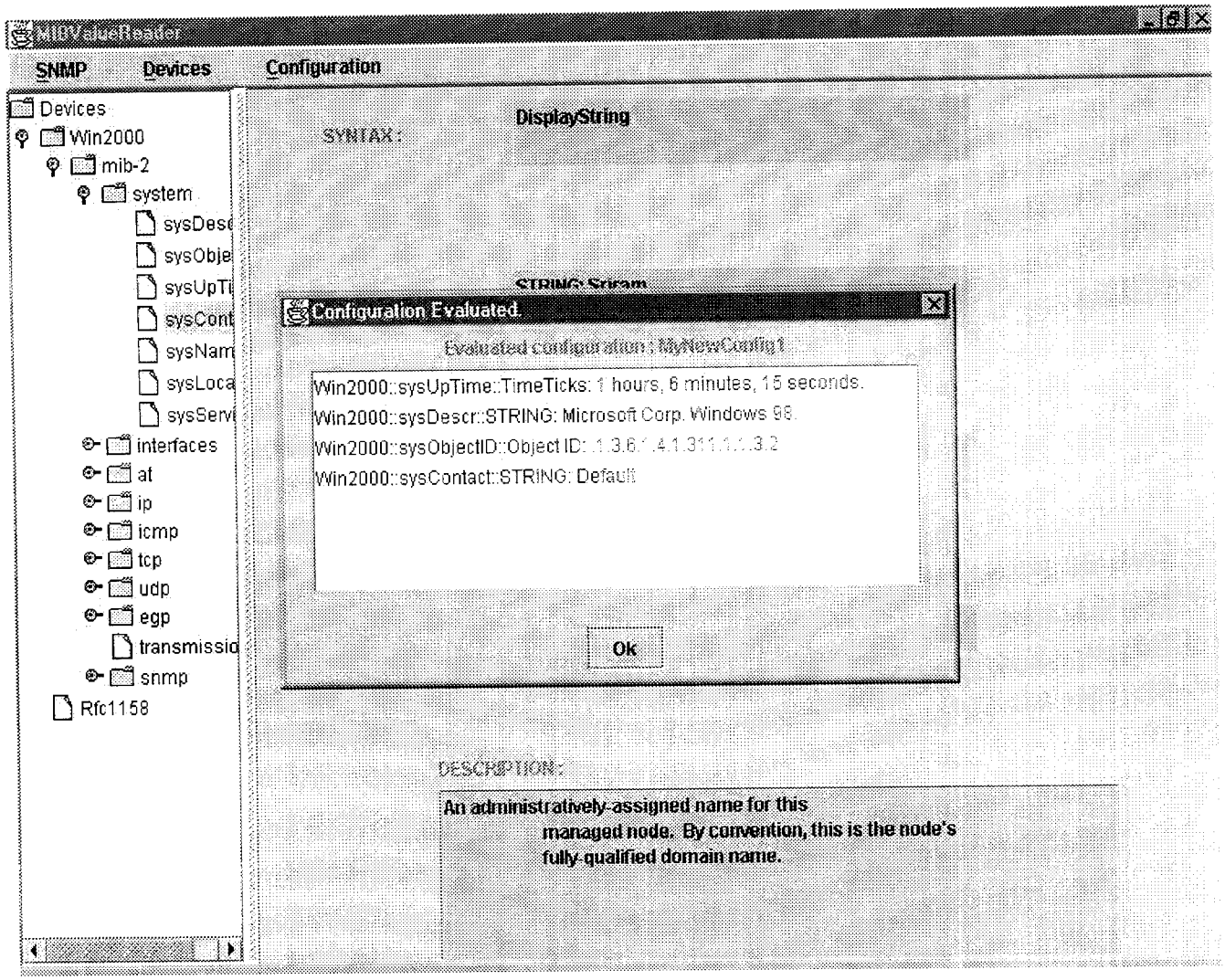

MIBValueReader
SNMP
Devices
Configuration
Devices
Win2000
mib-2
system
sysDesc
sysObje
sysUpTi
sysCont
sysNam
sysLoca
sysServi
interfaces
at
ip
icmp
tcp
udp
egp
transmissio
snmp
Rfc1158
SYNTAX:
DisplayString
VALUE:
STRING: Sriram
ACCESS:
OID:
DESCRIPTION:
An administratively assigned name for this managed node. By convention, this is the node's fully-qualified domain name.
Create New Configuration
Enter new Configuration file name
MyNewConfig1
Ok
Cancel

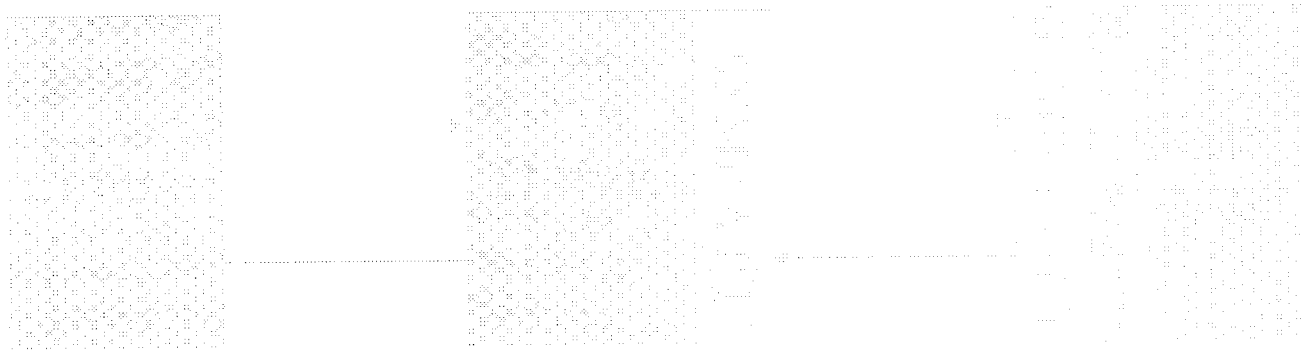

SNMP
Devices
Configuration
Devices
Win2000
mib-2
system
sysDes
sysObje
sysUpTi
sysCont
sysNam
sysLoca
sysServ
interfaces
at
ip
icmp
tcp
udp
egp
transmissio
snmp
Rfc1158
SYNTAX:
DisplayString
VALUE:
ACCESS:
OID:
Add OID to Configuration
Choose configuration to add to:
rama
rama1
MyNewConfig1
Add
Cancel
DESCRIPTION:
An administratively-assigned name for this
managed node. By convention, this is the node's
fully-qualified domain name.


MibValueHeader
SNMP
Devices
Configuration
Devices
Win2000
mib-2
system
sysDesc
sysObje
sysUpTi
sysCont
sysNam
sysLoca
sysServi
interfaces
at
ip
icmp
tcp
udp
egp
transmissio
snmp
Rfc1158
SYNTAX:
DisplayString
VALUE:
STRING: Sriram
ACCESS:
OID:
View Configuration
List of Configurations:
rama
rama1
MyNewConfig1
View
Cancel
DESCRIPTION:
An administratively-assigned name for this
managed node. By convention, this is the node's
fully-qualified domain name.


MIBVisualBrowser
SNMP
Devices
Configuration
Devices
Win2000
mib-2
system
sysDesc
sysObje
sysUpTi
sysCont
sysNam
sysLoca
sysServe
interfaces
at
ip
icmp
tcp
udp
egp
transmissio
snmp
Rfc1158
SYNTAX:
DisplayString
VALUE:
ACCESS:
OID:
Configuration Details
Contents of : MyNewConfig1
Win2000::sysUpTime
Win2000::sysDescr
Win2000::sysObjectID
Win2000::sysContact
Ok
Remove OID
DESCRIPTION:
An administratively-assigned name for this
managed node. By convention, this is the node's
fully-qualified domain name.


MibValueReader
SNMP
Devices
Configuration
Devices
Win2000
mib-2
system
sysDese
sysObje
sysUpTi
sysCont
sysNam
sysLoca
sysServ
interfaces
at
ip
icmp
tcp
udp
egp
transmissio
snmp
Rfc1158
SYNTAX :
DisplayString
VALUE :
Evaluate Configuration.
rama
rama1
MyNewConfig1
ACCESS :
OID :
Obtain Values
Cancel
DESCRIPTION :
An administratively-assigned name for this
managed node. By convention, this is the node's
fully-qualified domain name.


SNMP
Devices
Configuration
Devices
Win2000
mib-2
system
interfaces
at
ip
icmp
tcp
udp
egp
snmp
Rfc1158
DisplayString
SYNTAX:
Configuration Evaluated
Evaluated configuration: MyNewConfig1
Win2000::sysUpTime::TimeTicks: 1 hours, 6 minutes, 15 seconds.
Win2000::sysDescr::STRING: Microsoft Corp. Windows 98.
Win2000::sysContact::STRING: Default
Ok
DESCRIPTION:
An administratively-assigned name for this
managed node. By convention, this is the node's
fully-qualified domain name.



PRODUCT TESTING

All modules hereunder are tested based on the assumption that the RMI server which provides all functionalities required is up and responding.

Module 1: *Authentication*

The user is prompted for the administrator password which is already supplied with the product to the customer, which the administrator is allowed to change at the server.

If the correct password is typed the user has access to certain administrator privileges such as **Set**, **Add** and **Remove device.** A wrong password entered will only cause a repeat request for the correct password.

Bypassing authentication by cancelling the procedure allows the user access to the tool without the above-mentioned privileges. No other means of bypassing authentication is possible.

Module 2: *Get Operation*

The user has to select an OID from the tree and then request a Get. The server contacts the agent and obtains the value for the client.

The IP address of the agent has to be configured in the server correctly. If the server does not locate the agent at the specified address it indicates an error message.

The server requires the appropriate MIB file to be present failing which it indicates an error loading MIB file.

For a successful Get Operation the user has to select a leaf node from the device tree which indicates an OID failing which the user is prompted to select a proper OID.

Module 3: *Set Operation*

The user selects an OID from the device tree and then requests a set operation. The user is prompted for a new value and type for the particular OID. These are in turn received by the server which performs the set operation in the device.

A wrong type specification for an OID will not result in a successful set.

The server requires the appropriate MIB file to be present failing which it indicates an error loading MIB file.

For a successful Set Operation the user has to select a leaf node from the device tree which indicates an OID failing which the user is prompted to select a proper OID.

Module 4: *Devices*

The administrator is allowed to add or remove devices for monitoring.

When adding a device the user is prompted for a device name and its IP address. These are saved in the server.

Module 5: *Configurations*

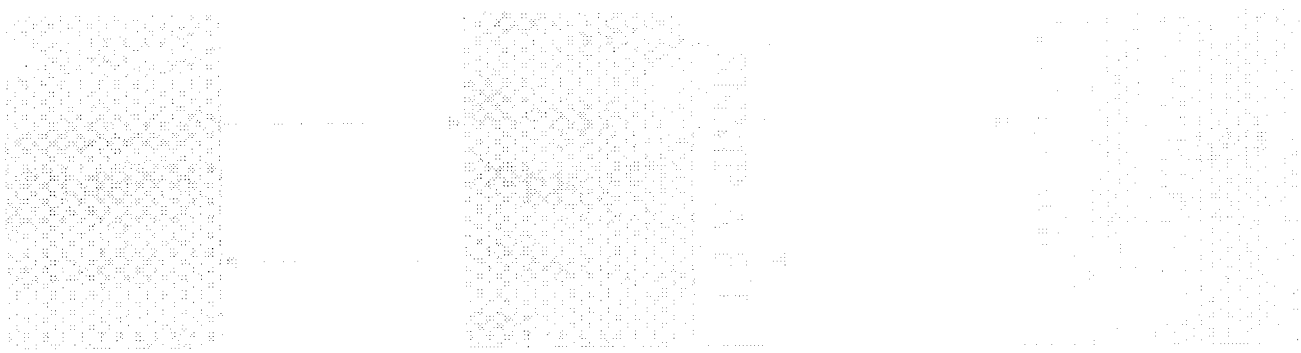
The user is allowed to create configurations to enable getting of multiple OIDs at the same instant. If the existing name of a configuration is specified the user is prompted for a new name. OIDs from several devices may feature in a single configuration.

An OID may be selected on the device tree and added to an already existing configuration. The user may not perform the add-to-configuration operation without selecting an OID.

The user is also allowed to view the contents of a configuration. The operation fails with an error message if a configuration has not been selected. Further, from the contents of a configuration, the user has the option of selecting and removing an OID.

An entire configuration may be selected for evaluation. The server performs a get for every OID in the configuration and returns to the client a complete list of values. An error in any one of the get operations is reflected at the client.

An entire configuration may be selected and deleted. An error message will be displayed if no configuration is selected.



FUTURE ENHANCEMENTS

We intend adding the following extra functionalities to the project.

- **Setting alarms:**

This is a feature implemented at the server and carried out by the server. The motive of alarms is to enable the user to identify imminent failure situations and take corrective actions before actual failure occurs. Here the server automatically gets the value for the OID(s) for which an alarm is set at specified intervals of time and checks for specified boundary conditions and prompts the client with an alarm if necessary.

- **Handling of SNMP traps:**

The client shall have the option of listening to traps generated by the agent through the server.

- **Dynamic Detection of new devices to the network**

The application will be able to dynamically detect the devices connected to the network which is to be monitored and managed.

CONCLUSION

The sole purpose of this project was to learn the industry technologies used in the production of client/server software of this kind and to create a software from the knowledge gained.

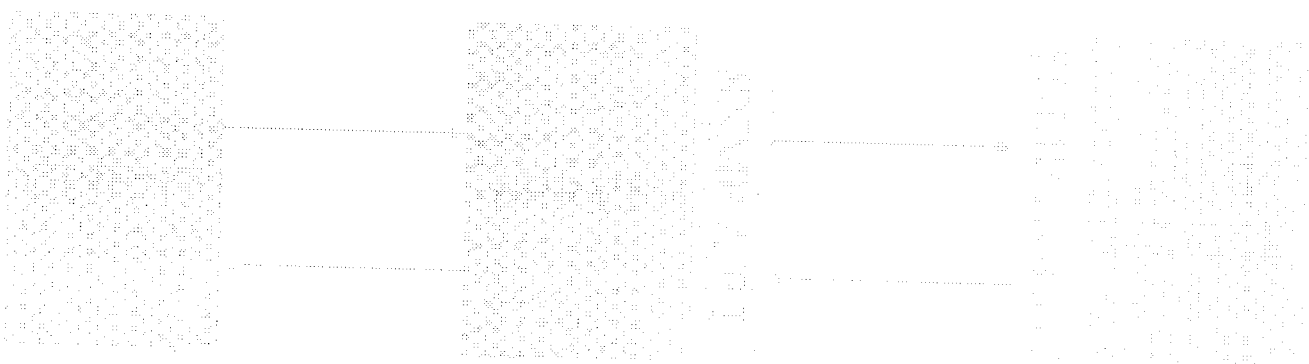
The MIBValueReader was chosen as the project in order to provide a tool free of cost for the use of network administrators and in this aspect, we consider having accomplished what we set out to.

We accept that several enhancements and up gradations are required and assure that some of them are already in the pipeline.

The MIBValueReader project to read MIB values at run time from SNMP compatible devices is successfully completed. Also added features include setting of SNMP values and configurations.

There is scope for future enhancements as is already specified in this document.

- www.adventnet.com
- www.rad.com
- www.apache.org
- William Stallings - SNMP, SNMPv2, SNMPv3 and RMON1&2, third edition, Pearson Education Asia.
- Herbert Schildt – Java2 The Complete Reference, fourth edition. Tata McGraw-Hill.
- Complete Java2 SDK documentation at www.java.sun.com
- www.microsoft.com
- www.freesoft.org
- www.snmplink.org



BIBLIOGRAPHY