



## **Configuration Wizard for a Broadband Access Server (BAS) AXC 706 (Tigris)**

Master Thesis Report Sherwin Aghilipour Engineering Physics Royal Institute of Technology KTH	April 2001
--	------------

Examiner: Björn Pehrson Department of Microelectronics and Information Technology KTH	Supervisor: Vladimir V. Vlassov Department of Microelectronics and Information Technology KTH
--	--

Supervisor at Ericsson: Gunnar Axelsson Broadband Access & VPN Solutions Ericsson Telecom AB
---

Master Thesis Report April 2001 .....	1
Shervin Aghilipour .....	1
<b>1. Abstract .....</b>	<b>4</b>
<b>2. Introduction.....</b>	<b>4</b>
2.1 Feature summary .....	5
<b>3. Network Overview .....</b>	<b>5</b>
3.1 Broadband.....	6
3.2 ATM Technology.....	6
3.2.1 ATM Cell Basic Format.....	7
3.2.2 ATM Services .....	7
3.2.3 ATM Virtual Connections.....	7
3.3 Network Scenarios .....	8
3.3.1 PPP over ATM.....	9
3.3.2 Point to Point Tunneling Protocol (PPTP) .....	9
3.3.3 Network Address Translation (NAT).....	11
<b>4. Access Setup Wizard Overview .....</b>	<b>12</b>
4.1 ASW User Interface.....	13
4.2 GUI Framework.....	15
4.2.1 The View Panels .....	16
4.2.2 Command Panel .....	17
4.2.3 Configuration Panel .....	18
4.2.4 ISP Settings Panel .....	19
4.2.5 Access Method .....	21
<b>5. Command Line Interpreter.....</b>	<b>22</b>
5.1 Introduction to Command Usage.....	24
5.1.1 Access Partition.....	24
5.1.2 Service Profile.....	25
5.1.3 Configuring the ATM Interface .....	26
5.1.4 RADIUS Authentication and Accounting.....	29
<b>6. Structure of Access Setup Wizard ASW .....</b>	<b>30</b>
6.1 The Flow Diagram of the Handler class.....	32
6.2 Create ISP.....	33
6.3 PPP over ATM Configuration.....	34
<b>7. Conclusions .....</b>	<b>35</b>
<b>8. References .....</b>	<b>36</b>
<b>9. Additional Information Sources .....</b>	<b>37</b>
<b>Appendix A.....</b>	<b>38</b>

<b>Appendix B</b> .....	<b>39</b>
<b><u>The Handler class</u></b> .....	<b>39</b>
<b><u>The lsp class</u></b> .....	<b>59</b>
<b><u>The PPPoATM class</u></b> .....	<b>62</b>
<b><u>The lspObj class</u></b> .....	<b>69</b>

# Configuration Wizard for a Broadband Access Server (BAS) AXC 706 (Tigris)

## 1. Abstract

The configuration of an access server are often a complex and time-consuming experience, the risk of excluding commands or creating other incorrect configurations details are not uncommon. Corrections to these problems consume time that can be better used elsewhere on other more important issues, any tool that can aid a person with the problem will only help speed the access server(s) into commissioning or back into service again. Developing a software configuration tool is one way of eliminating such unnecessary experiences.

It is a complex task to correctly configure a BAS (Broadband Access Server) in a network scenario. The user must know exactly how the network looks like, and have knowledge about IP addressing, ATM PVC's, etc. The user must also know how to configure the BAS. The scope of this Master thesis work would be to create a Java program that asks accurate, intelligent questions to the user and with this information create a working configuration file for the BAS. The configuration file, which is a script file (file.scr), will be loaded to the flash memory of the BAS.

The program only assumes that the user have knowledge of how the network looks like. The specific commands for the BAS, and which order they are put in, are totally handled by the program. The program could also educate the user of how certain command works.

## 2. Introduction

A great variety of access technologies and increasing requirements on performance, as a result of the success of the Internet and Web-based computing, new access and edge nodes are necessary. The Ericsson's AXC 706 Broadband Access Server (BAS) will offer access flexibility with a high level of functionality, allowing operators to provide IP access solutions to support their particular service requirements, access technologies and backbone network architectures.

The AXC 706 will function as a Broadband Access Server (BAS) providing access to ADSL (Asymmetric Digital Subscriber Line) and Local Area Network (LAN).

It will be also developed into a true Multiservice access platform supporting current narrowband applications such as dial-in modem access and ISDN (Integrated Services Digital Network).

## 2.1 Feature summary

Features of the AXC 706 are:

- A complete set of routing protocols
- PPP (Point to Point Protocol)-based access over all media
- Full RADIUS (Remote Authentication Dial-In User Service)
- ISP (Internet Service Provider) selectivity
- Multi-protocol Label Switching (MPLS)
- Label Edge Router (LER) support
- Bridged Ethernet access possibilities
- Tunneling of PPP across the network (both L2TP and PPTP) for Virtual Private Networks (VPN). [10]

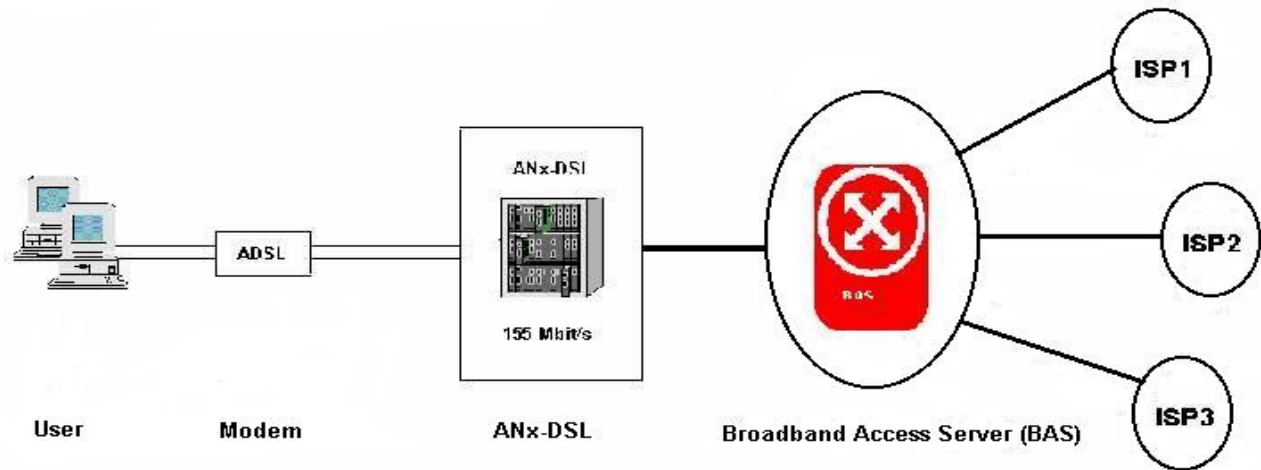
## 3. Network Overview

The AXC 706 Broadband Access Server, integrated into the ANx (Ericsson Broadband Access Network) system, is connected to the rest of the network via one of the network interfaces of the ANx system.

The ANx product enables network operators to provide broadband services using existing copper lines. The system uses Asymmetric Digital Subscriber Line (ADSL) Technology, which enables traffics up to 8 Mbit/s and transfer the digital information as Asynchronous Transfer Mode (ATM) cells.

Edge Router (ER) is connected to the ANx access network in order to provide the necessary TCP/IP connection between the connected subscribers and the Internet Service Providers (ISPs).

A typical network scenario is shown in figure 3.1.



**Figure 3.1 Network Overview, user is connected by ADSL modem to the Internet Service Provider (ISP) via Anx-DSL and router (BAS).**

### 3.1 Broadband

Asymmetric Digital Subscriber Line (ADSL) is such a broadband service, which is available for connecting to the network via the local telephone exchange.

The ADSL central concentrator will be connected to the access server through the ATM link.

The user is connecting their PC's by some dial-up networking software to get connection to a service provider and authentication is provided by a RADIUS server in the network which supplies username and password details.

When authentication is successfully done, the user will be connected to the Internet or there corporate network with a typically ADSL speed up to 8 Mbit/s which is an optimal access rate and it is much higher than Narrowband infrastructure. [1]

### 3.2 ATM Technology

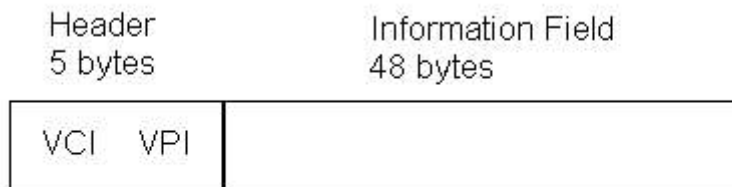
Asynchronous Transfer Mode (ATM) is an international standard for cell relay information multiple service types, such as voice, video, or data, packed in small, fixed-sized cells.

ATM is a cell-switching and multiplexing technology that combines the benefits of circuit switching with those of packet switching, and its bandwidth rate is from a few megabits per second to many gigabits per seconds.

Because ATM is asynchronous, the time-division is available for identifying the source of the transmission, which is in the header of each ATM cell. This can be described as real-time behavior of the data transmission.

### 3.2.1 ATM Cell Basic Format

ATM transfers information in fixed-size units called *cells*. Each cell consists of 53 bytes. The first 5 bytes contain cell-header information, and the remaining 48 contain user information field. Small fixed-length cells are well suited to transferring voice and video traffic because such traffic is intolerant of delays that result from having to wait for a large data packet to download. [2]



**Figure 3.2 ATM cell header showing Virtual Path Identifier (VPI) and Virtual Channel Identifier (VCI)**

### 3.2.2 ATM Services

Three types of ATM services exist: Permanent Virtual Connections (PVCs), Switched Virtual Circuits (SVC), and connectionless service.

A PVC is logical (rather than a physical) connection between two communicating ATM peers. Such a connection is typically established by a network administrator.

A PVC is typically used for interconnectivity between fixed corporate locations, data centers, or regional hubs engaged in traditional data communications.

The user can easily choose or change PVCs, because this kind of connection is static.

A SVC is created and released dynamically and remains in use only as long as data are being transferred. You can compare it as a telephone call. Dynamic control call requires a signaling protocol between the ATM endpoint and the ATM switch. [4]

### 3.2.3 ATM Virtual Connections

Two types of ATM connections exist: *Virtual Path (VP)*, which are identified by *Virtual Path Identifier (VPI)*, and *Virtual Channels (VC)*, which are identified by the combination of a VPI and a *Virtual Channel Identifier (VCI)*.

A virtual path is a bundle of virtual channels, those which are switched across the ATM network, and permanently connected two points together.

The virtual channels are a logical subdivision of the virtual path, and have only local significance across a particular link.

Each virtual path (VP) can contain number virtual channels (VC), and both virtual

Path and virtual channels can be distributed over a number of nodes.

A virtual path is normally set up by a service provider through a management system used to manage the operator ATM network.

Virtual channel connections can be set up dynamically, also by the operator.

Figure 3.3 illustrates how VCs concatenate to create VPs, which, in turn, concatenate to create a transmission path (a physical link). [3]



**Figure 3.3 VC concatenate to create VPs**

### **3.3 Network Scenarios**

The AXC 706 Broadband Access Setup Server (BAS) is well suited for concentrating traffic coming from an ATM network. The justification for locating the BAS directly in the ATM network exists where there is a wide geographical distribution of ATM connected users. The BAS can support up to 2000 PVCs per ATM card.

In Figure 3.4, the BAS is connected to an Ericsson Anx-DSL concentration shelf. Each customer would have an ATM PVC established to the BAS from their Network Terminal (NT). Bridge Ethernet is running over this. Depending on the configuration by the operator, the BAS may be providing access on behalf of a single ISP (Internet Service Provider), or alternatively providing ISP selectivity.

The user (administrator) can select different network scenarios in order to make configuration for desired connection to the ISP(s), such as PPP over ATM, PPTP (Point-to-Point Tunneling Protocol), and NAT (Network Address Translation).



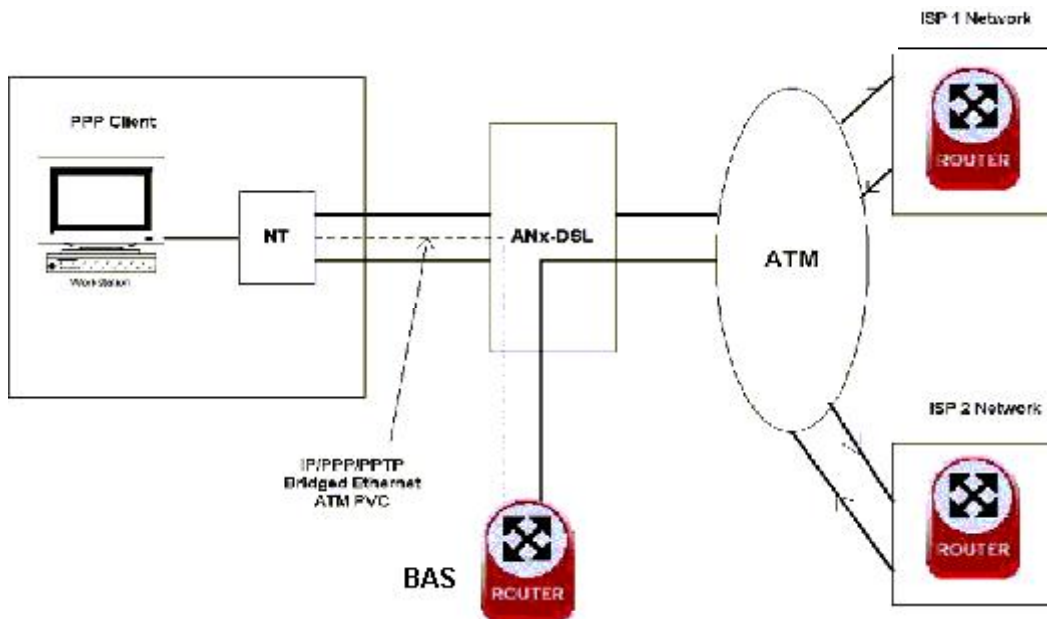


Figure 3.4 Link from BAS to ISP

### 3.3.1 PPP over ATM

Point to Point Protocol (PPP) is a protocol that supports dial-in, providing a point to point connection between two different TCP/IP systems for the transfer of IP datagrams.

PPP is a datalink protocol that encapsulates IP datagrams and carries them over serial lines. PPP has been designed to operate over both asynchronous connections and bit oriented connections and it is compatible for accessing the ISPs. PPP over ATM enables PPP connection directly over the ATM PVC via ADLS link. Alternatively, we can just install an ATM Network Interface Card (NIC) into the PC, providing a connection to the ATM port in the Network Terminal (NT). In this way, the PPP is carried over ATM directly from the PC to the router.

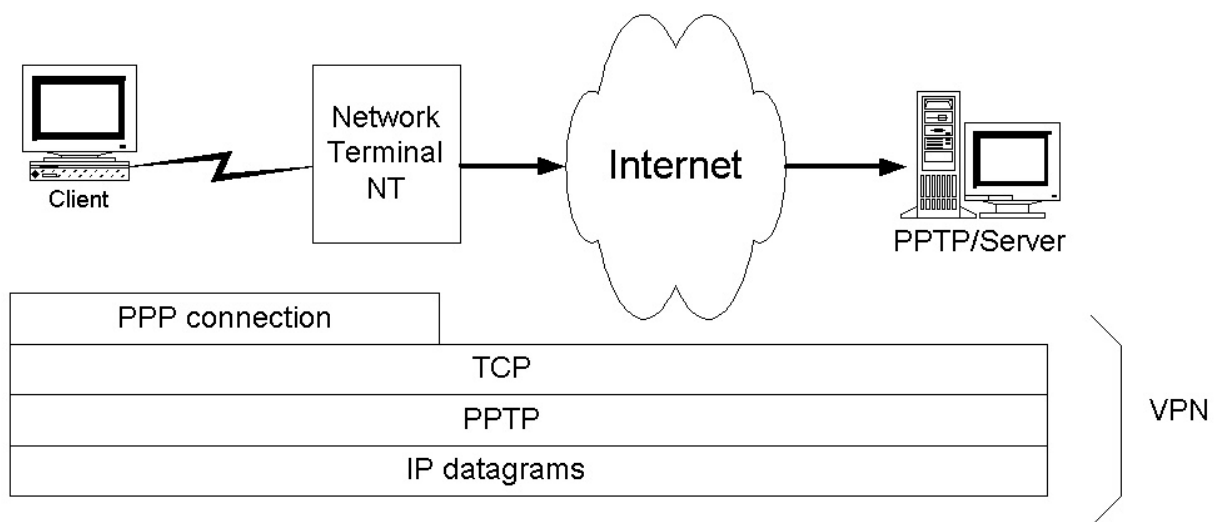
A disadvantage of this scenario is that only a single PPP session can be transmitted over each PVC. If we desire to connect more than one user, extra PVC would be required per user. This takes more resources and management issues.

### 3.3.2 Point to Point Tunneling Protocol (PPTP)

Point-to-Point Tunneling Protocol (PPTP) is a network protocol that enables the secure transfer of data from a remote client to a private enterprise server by creating a Virtual Private Network (VPN) across TCP/IP-based data networks.

The PPTP protocol is used to create dynamic tunnels in the access network. Tunneling is a method for encapsulating packets inside a protocol that is understood at the entry and exit point of the network. This protocol enables PPP session of remote clients to be tunneled across public network to ISPs or corporate networks so that the remote clients appear to be connected locally. The entry and exit points of the network are defined as tunnel interface.

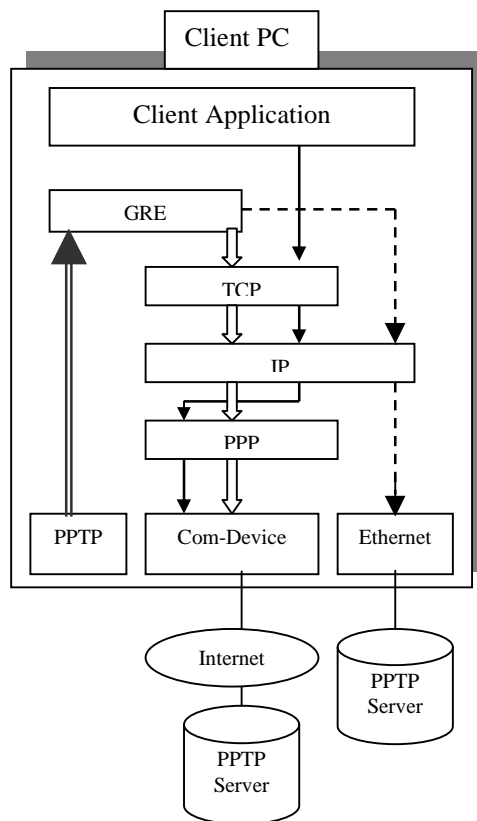
Generally, after the client has been connected to the ISP through PPP connection, a second network layer is made over the existing PPP connection. Data sent using this second connection has the same form as the IP datagrams, which contain encapsulated PPP packets. The second layer crates the VPN connection to a PPTP server as a tunneled connection. Figure 3.5 illustrates this scenario.



**Figure 3.5 The PPTP Tunnel**





Tunneling enables the router to connect to the private networks. The packets, which are sent to the PPTP server via router, will be transferred to the destination computers through the VPN. The encapsulated PPP packet contains multi-protocol data such as TCP/IP, IPX or GRE (Internet Generic Routing protocol).

PPTP packets from a remote access PPTP client and a local LAN PPTP client are processed differently. A PPTP packet from a remote access PPTP client is placed on the telecommunication device physical media, while the PPTP packet from a LAN-PPTP client is placed on the network adapter physical media as illustrated in the follow figure:



**Figure 3.6 Placing a PPTP packet on the Network Media**

The figure above illustrates how PPTP packets and then places the outgoing PPTP packet on a modem, ISDN, or LAN network media.

- PPTP encapsulation      
- Outgoing packet        
- to remote network      
- to LAN                    

Virtual ports are used to create an interface between ATM and a dynamic PPTP tunnel, the maximum number that can be created is 2000. The virtual port is bound to a tunnel session whenever a new session is established. This binding of a virtual port to a tunnel session is removed as soon as the session disconnects. [6, 12]

### 3.3.3 Network Address Translation (NAT)

The Internet is based on 32-bit IP (Internet Protocol) addresses, which means the theoretical maximum number of computers on the Internet is 4 billion or so. The practical limit is much lower. In fact, the Internet may be only a few years away from running out of IP addresses.

The NAT (Network Address Translation) was developed to allow the use of a single IP address for a whole network or computer.

A NAT is sitting between the public Internet and the network, and what it does is to rewrite the IP addresses and port numbers in IP headers so the transferring packets appear to be coming from a single public IP address.

NAT allows the recycling of routable address classes by translating non-routable Intranet addressing schemes into routable, globally unique addressing schemes. NAT is transforming the incoming IP packets in the way that, it translates the IP address in the header and replace it with the new IP addressing scheme, which is required by the Internet host system.

## 4. Access Setup Wizard Overview

Access Setup Wizard (ASW) is a Java-based Graphical User Interface (GUI) to enable configuration for the most common features of the BAS (Broadband Access Server, Tigris).

The user will be able to configure access server RADIUS, ATM and IP settings using the graphical aid and can rapidly establish basic network connectivity.

The network control center can be connected and assist in the more complex network features such as ISP service selection configuration.

The user should only know how the network is build up and what network scenario is attempting to be used. The user can easily fill in the right data, such as IP address, IP gateway, PVCs, and save the configurations into a single script file.

After making configuration the ISP(s) and desired network scenario, the script file can be saved by the user.

A complete configuration script may be generated in ASW without uploading it to the Tigris. The user has possibility to view or edit the script file in a separate window.

When the script is satisfactory it can be uploaded to the Tigris.

An entire configuration script can be written to the flash memory of the router and it will be saved there for later use, or removed it by a new file.

The content of the script file could be varied depending on what network scenario is used. It is also possible to make a single configuration that cover different scenarios like Network Address Translation (NAT), Point to Point Tunneling Protocol (PPTP) and PPP over ATM.

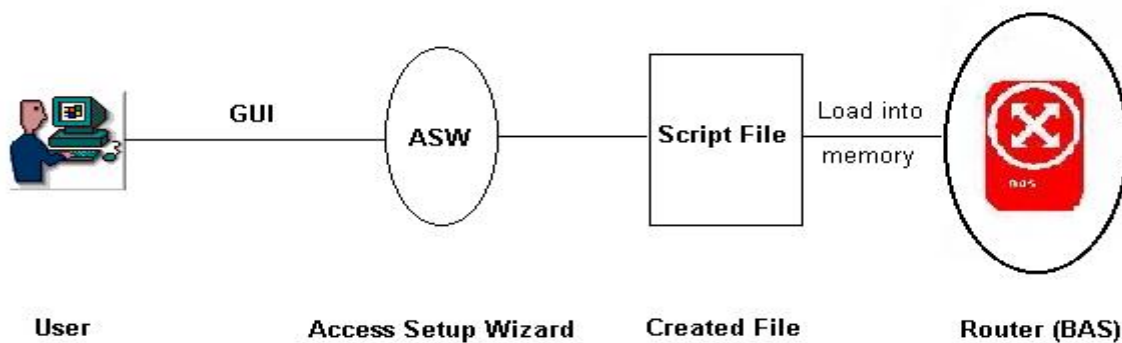
This program creates the script file in very short time and can be loaded into the flash memory of the router by the user. Whiteout using this program, the user has to write all commands, line by line.

An example of the configuration script is shown as below, and will be explained in chapter 5 (Command Line Interpreter).

```

SET PROMPT "BAS_706$CONTEXT$SYNC$SAVED"
SET VIRTUAL PORT COUNT 10 0 10 10 10 10 10
ADD ACCESS PARTITION ENTRY "ISP1" "Gate One"
SET ACCESS PARTITION IP GATEWAY "ISP1" 192.168.221.60
SET DHCP SERVER START ADDRESS V6.1 192.168.231.1
SET DHCP SERVER END ADDRESS V6.1 192.168.231.10
ADD ATM INTERFACE VIRTUAL CHANNEL PVC J1 0 100 1 1 llc 9188
SET VIRTUAL PORT PHYSICAL PORT V4.1 J1
ADD ATM PPP PVC V4.1 0 45
ADD ATM CIP_PVC PVC V7.1 0 100
CONTEXT ISP1
ADD IP NETWORK ENTRY 192.168.221.40 255.255.255.0 V7.1
SET IP NETWORK MTU 192.168.221.40 9180
ADD IP ROUTE ENTRY 192.168.213.0 255.255.255.0 192.168.221.60 1
ADD IP ROUTE ENTRY 192.168.209.0 255.255.255.192 192.168.221.60 1
ADD RADIUS AUTHENTICATION SERVER ENTRY 5 192.168.213.1 "edge" 5 3 1645
ADD RADIUS ACCOUNTING SERVER ENTRY 5 192.168.213.1 "edge" 5 3 1646

```

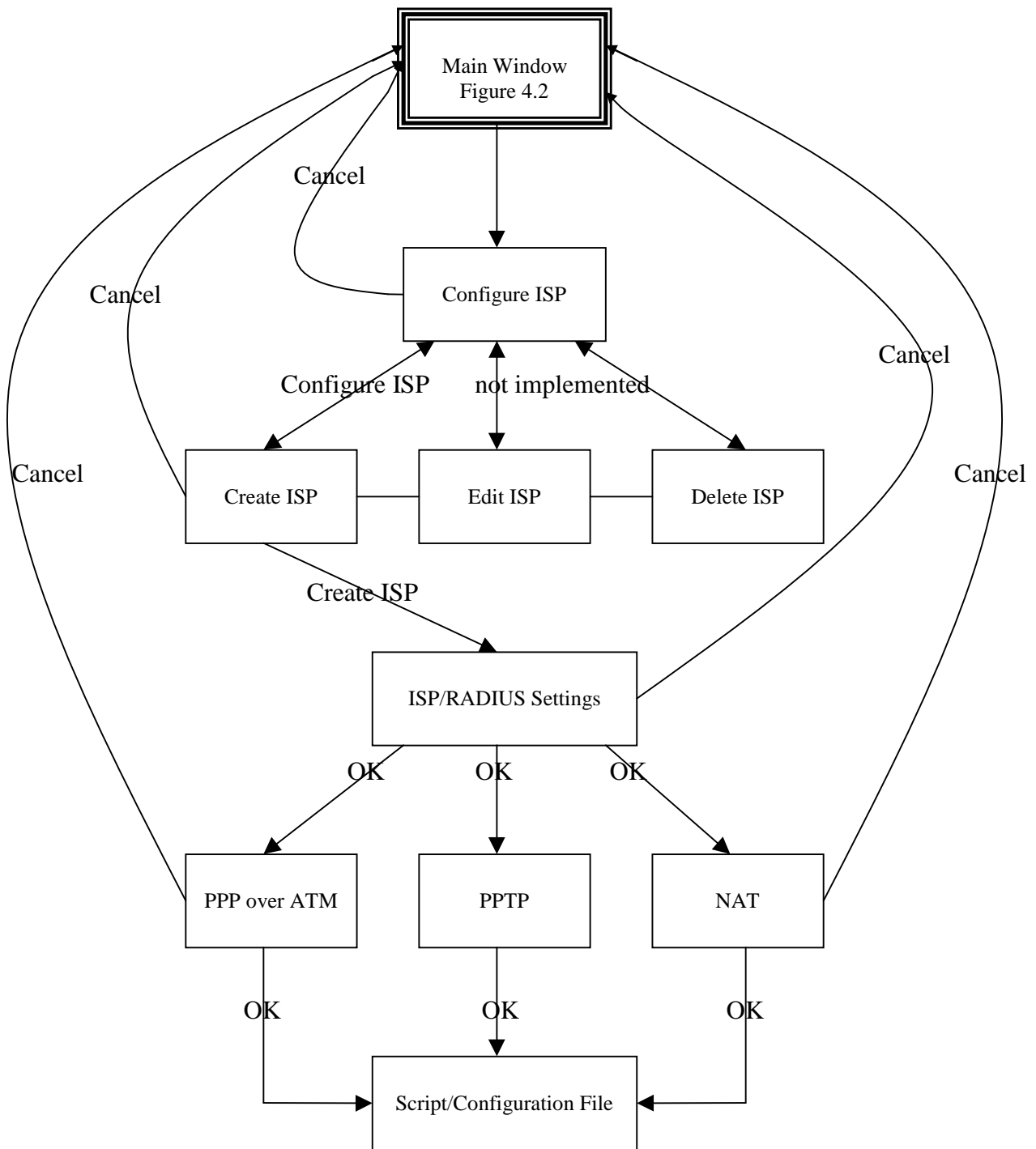


**Figur 4.1 User cretaes the script file by using ASW that will be loaded into the router**

## 4.1 ASW User Interface

The ASW user interface consists of a main configuration window with panes offering a series of choices. Each pane contains different fields, which needs to be filled in by the user. The user has the option of changing or editing the information in order to make a new configuration.

The diagram 4.1 illustrates roughly, connection between the windows. It shows only a simple overview of the flow.

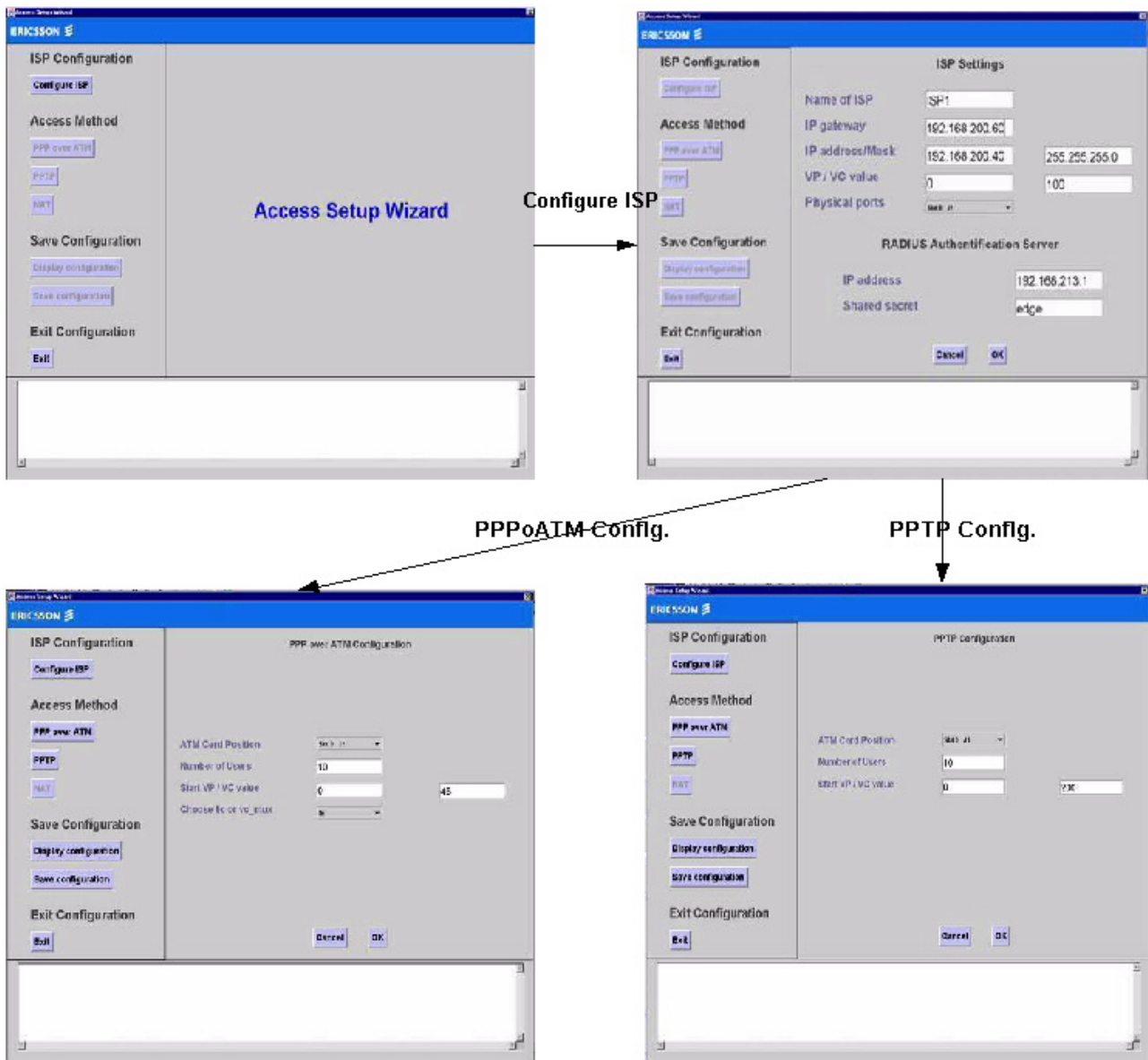


**Diagram 4.1 Links between the Windows**

## 4.2 GUI Framework

The GUI Framework consists of different visual parts (standardized GUI panels). Each panel is showing the most particular information, which is needed to configure a correct network scenario. The user can just easily browse through the panels in order to make configuration for the ISP(s) and access methods.

The sequence of these configuration panels is shown in Figure 4.2.



Figur 4.2. The sequence of configuration panels, shown by the ASW

## 4.2.1 The View Panels

A hierarchical panel simplifies the organization of information. The main screen consists of different parts as follow:

Command Panel, Configuration Panel and Text Area.

The main screen is illustrated as in Figure 4.3.

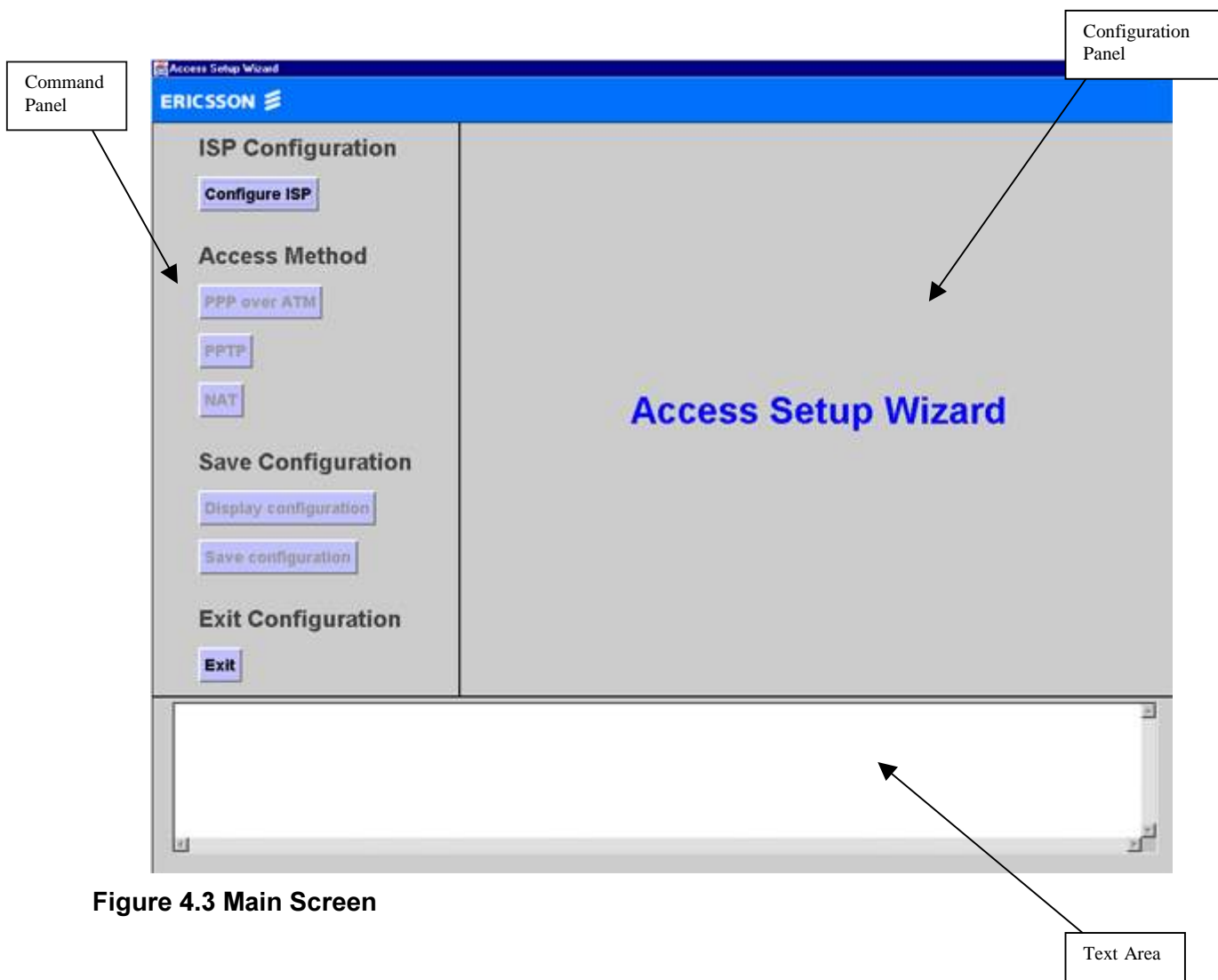


Figure 4.3 Main Screen



## 4.2.2 Command Panel

This panel is divided into different sections, which gives selectivity of particular configuration. Each section consists of *buttons*, which can be selected.

### ◆ *ISP Configuration*

This section is used for making configuration of ISP(s). The user should choose to configure one or several ISP(s) with appropriate information, such as IP address, IP gateway and VP/VC-values.

### ◆ *Access Method*

After selecting an ISP, the user needs to choose an access method.

This means that a network scenario (PPP over ATM, PPTP or NAT) should be selected.

These buttons are initially inactivated as long as any ISP configuration has not been made.

The user must configure an ISP before using any access method.

### ◆ *Save Configuration*

This section gives the user opportunity to see and analyze the configuration by selecting *Display Configuration*.

In this case all command lines will appear in the text area below the screen.

The commands give information, how the network is configured and if there is any incorrect data, the user could select new configuration before saving it to the script file by selecting the *Save Configuration* button.

### ◆ *Exit Configuration*

This option exits the configuration and closes the program.

If the user needs to make new configuration, the program should be started again.

The Command Panel area will always appear regardless of which action has been made.

Selectivity of buttons depends on what part of the configuration is in process.

When the program is started, only two opportunities is available, ISP configuration or exit the program. Access methods will be activated after selecting an ISP.

### 4.2.3 Configuration Panel

Configuration Panel is the area used for presenting options to configure ISP(s) and access methods. Different panels will show up by choosing one of the options from the Control Panel.

When the user selects to configure an ISP, the panel containing information for ISP configuration, appears as illustrated in Figure 4.4.

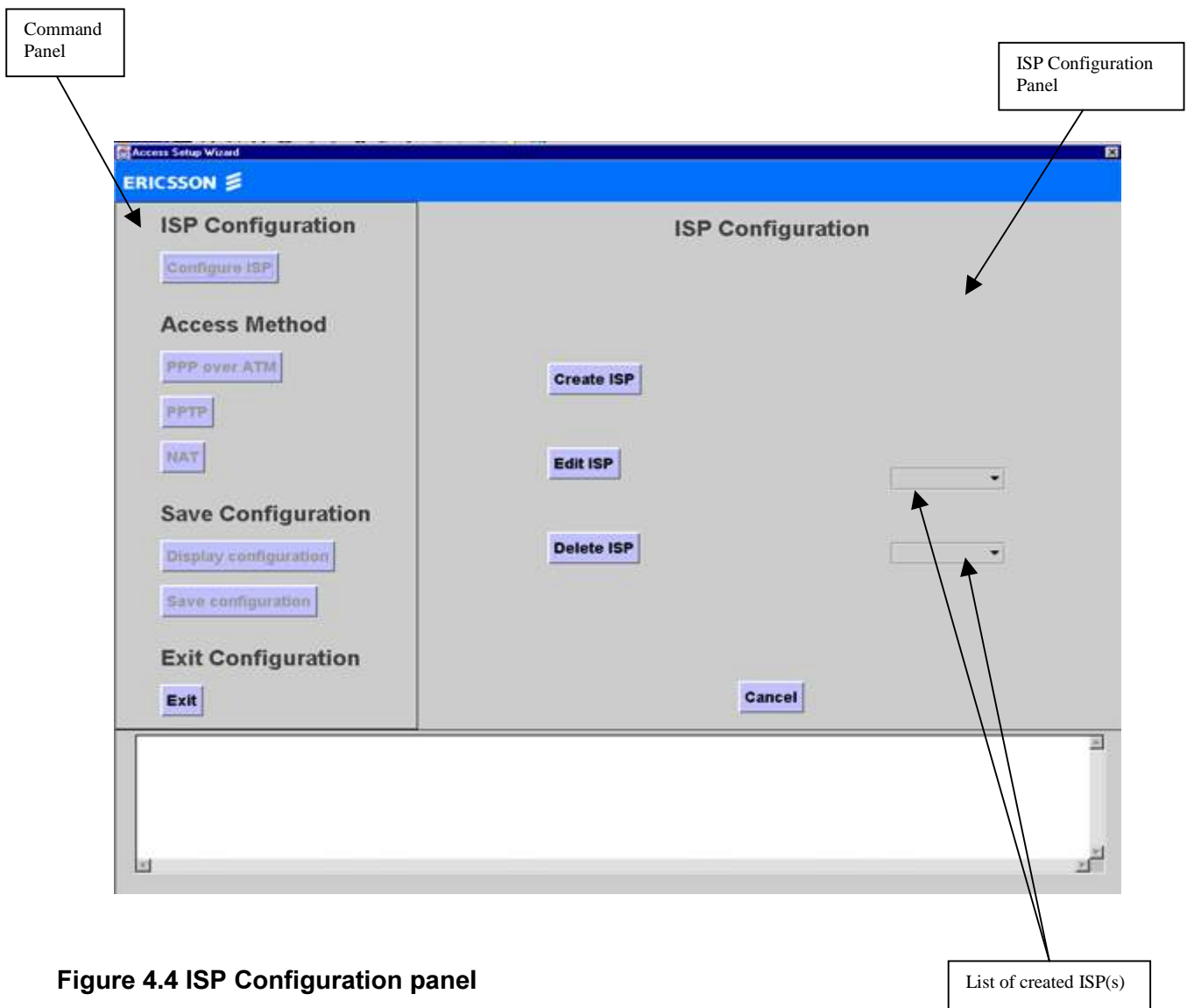


Figure 4.4 ISP Configuration panel

The options in this panel are:

- *Create ISP* button leads the user to the next panel, which contains the information for configure and create an ISP.
- *Edit ISP* button is used for editing or changing information for an existing ISP from the list of created ISP(s).
- *Delete ISP* button is used to delete ISP(s) from the list.
- *Cancel* button exits the configuration and shows the previous panel

## 4.2.4 ISP Settings Panel

When the user selects to create an ISP, then this panel will be appears as illustrated in Figure 4.5.

The screenshot shows the 'Ericsson Access Setup Wizard' window. The main area is titled 'ISP and RADIUS Settings Panel'. It is split into two columns. The left column contains 'ISP Configuration' with buttons for 'Configure ISP', 'Access Method' (PPP over ATM, PPTP, NAT), 'Save Configuration' (Display configuration, Save configuration), and 'Exit Configuration' (Exit). The right column contains 'ISP Settings' with fields for 'Name of ISP' (ISP1), 'IP gateway' (192.168.200.60), 'IP address/Mask' (192.168.200.40 / 255.255.255.0), 'VP / VC value' (0 / 100), and 'Physical ports' (set J1). Below this is the 'RADIUS Authentication Server' section with fields for 'IP address' (192.168.213.1) and 'Shared secret' (edge). At the bottom are 'Cancel' and 'OK' buttons. Three callout boxes with arrows point to the main panel, the 'Physical ports' dropdown, and the 'RADIUS Authentication Server' section.

Figure 4.5 ISP and RADIUS Setting Panel

All the fields shown in the figure 4.5 must be completed, otherwise a pop-up message will come up telling the user to fill all empty fields.

In order to create an ISP, the user should know how the network is configured. It means that the choice of *Physical Ports*, *PVCs* and *IP addressing* is the essential part of the configuration.

Suppose that the user wants to create an ISP called **ISP1**.

An *IP gateway* is needed to access the ISP or corporate network.

The user have been designated an IP address from a private address pool from there access provider and this will be required to access the ISP.

A list of *physical ports* (Slot0 J1, Slot1 J2, Slot2 J3, and Slot3 J4) is available to choose for accessing the LAN or ATM operation.

The physical port *Slot0 J1* indicates that the ATM card is placed in the first position of the router (Tigris). In this case there are four positions available for ATM card.

The ISP or corporate networks is connected through specific paths (ATM VCs). If it is the first ISP to be created, then the user should select appropriate PVCs, i.e. VPI and VCI value.

The same procedure can be done for making more than one ISP, but the user must be aware of not using the same PVC for different ISPs, because each ISP has a unique PVC.

The program has capability to know if there is any PVCs already occupied and in this case it will show up a pop-up message reminding the user to select another PVC.

The router will be directly connected to the network with a unique IP address and the network mask which is characteristic for the network, so when it routes, it knows which networks are local.

When IP runs over an ATM network, routers circle the edge of the ATM cloud.

Each router communicates with every other router by a set of PVCs configured across the ATM physical topology. The routers do not have direct access to information concerning the physical topology of underlying network, they have knowledge only of the individual PVCs that appear to them as simple point-to-point circuits between to routers.

*RADIUS (Remote Authentication Dial-In User Service)* is used to authenticate user through a series of communications between the client and the server. Once a user is authenticated, the client provides that user with access to the appropriate network services. RADIUS is a system of distributed security that secures remote access to networks and services against unauthorized access. RADIUS includes an authentication server and client protocols.

RADIUS authentication is used in both fixed ISP connectivity and selectable ISP connectivity solution that the router offers.

RADIUS server information, such as IP address must be available in order to get contact with server. For this reason, there is an option in the RADIUS setting panel that the user has possibility to get the specific RADIUS server IP address.

When ISP- and RADIUS configuration has completed, the information will be saved into a script file by pressing the *OK* button. At the same time the previous panel will appear and the user will be demanded to choose an *Access Method*.

At this moment, all three buttons (PPP over ATM, PPTP and NAT) from *Command Panel* are activated.

## 4.2.5 Access Method

Access Method is the way to configure a network scenario. PPP over ATM, PPTP and NAT are just among many other possible scenarios.

After a successful configuration of ISP(s), the user may want to establish a network scenario. By choosing an option, next panel will appear.

Figure 4.6 illustrates what information is needed for making a configuration for the scenario *PPP over ATM*.

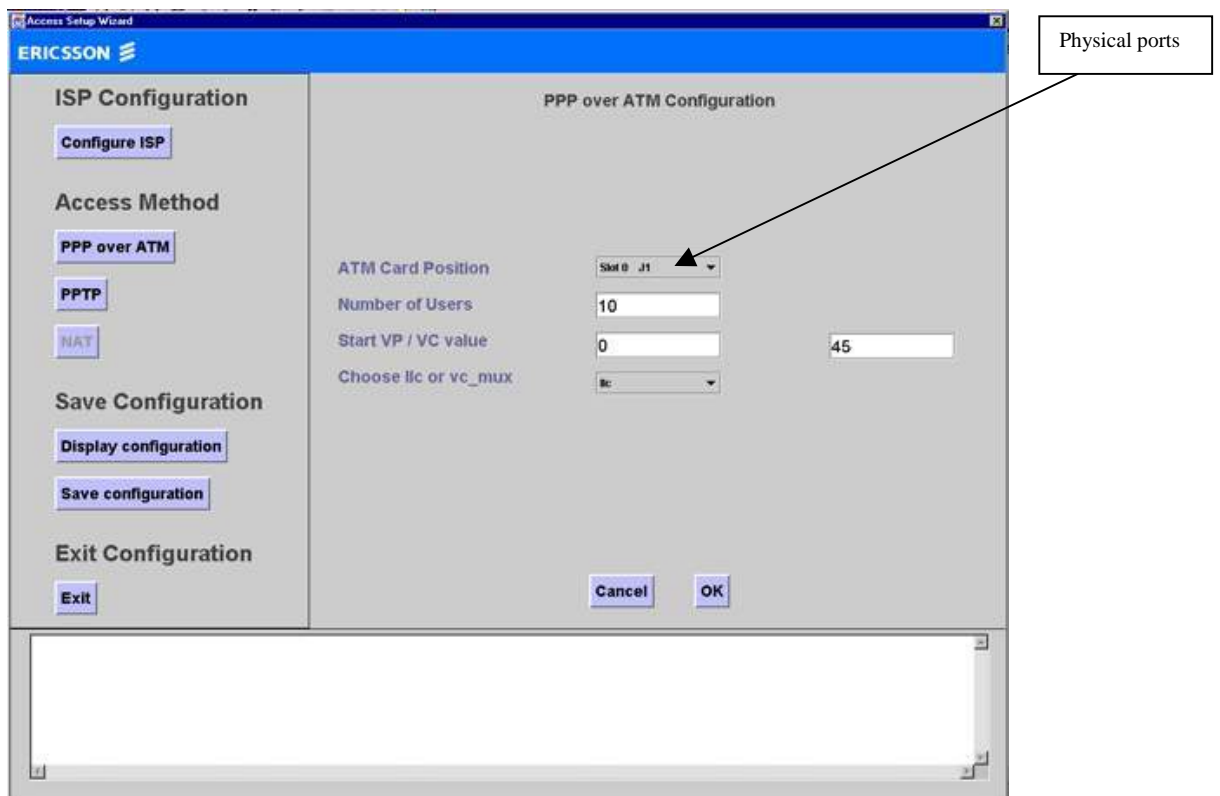


Figure 4.6 PPP over ATM Configuration Panel

In order to set up this scenario, we only need to know which ATM card is attempting to be used, how many users we are going to connect and which PVCs are selected for each user.

There are four ATM cards available and each card supports up to 2000 users, so the total number of users would be 8000.

As it is illustrated in the Figure 4.6 we attempt to configure the network for 10 users and the selected PVCs (VPI and VCI value) are 0/45.

The range of possible values is 0 to 255 for VPI (Virtual Path Identifier) and for VCI (Virtual Channel Identifier), the range is 32 to 65535.

So when we press *OK*, the program creates all necessary commands to connect all 10 users to the network, according to the selected network scenario (PPP over ATM).

At this step, we are almost done with the whole configuration and we can display or save this configuration to a file. We have also possibility to create another ISP(s) and select another access method and add the new information to the same file.

As it is mentioned before when we exit the program and run it again for making new configuration, the new information will overwrite the old configuration, so if the old file is still needed, it is recommended to keep a backup of all files.

The panels, which handle the configuration for other scenarios (PPTP and NAT), have almost the same interface as PPP over ATM panel. The only difference is that the program creates different command lines in the script file depending on the network scenario.

## 5. Command Line Interpreter

This chapter describes the contents of the script file, which is used to control and manage the Tigris (BAS) and offers some description of how the products are installed and operated.

Command line interpreter is the simplest interface that can be used to fully communicate with the access server. It is a text based command language and offers a management interface to add, delete, set and show parameters.

An example of configuration script file is shown below:

```
SET SCRIPT VERSION (Access Setup Wizard Version 5.1)
SET PROMPT "TIGRIS"
SET VIRTUAL PORT COUNT 10 0 0 0 3 0 0 1
ADD ACCESS PARTITION ENTRY "ISP1"
SET ACCESS PARTITION IP GATEWAY "ISP1" 192.168.221.60
SET RADIUS PORT COUNT 50
ADD SERVICE PROFILE ENTRY "SP_ISP1"
SET SERVICE PROFILE ACCESS PARTITION "SP_ISP1" "ISP1"
SET SERVICE PROFILE MODEM POOL "SP_ISP1" "default"
ADD SERVICE PROFILE ENTRY "ATMSELECT" CDNR
SET SERVICE PROFILE ACCESS PARTITION "ATMSELECT" "UID-SELECT"
SET SERVICE PROFILE MODEM POOL "ATMSELECT" "default"
SET SERVICE PROFILE PORT PROFILE V7.1 "SP_ISP1"
SET SERVICE PROFILE PORT PROFILE V4.1 "ATMSELECT"
SET SERVICE PROFILE PORT PROFILE V4.2 "ATMSELECT"
SET SERVICE PROFILE PORT PROFILE V4.3 "ATMSELECT"
ADD ATM TRAFFIC DESCRIPTOR 1 UBR 14800000
ADD ATM INTERFACE VIRTUAL CHANNEL PVC J1 0 45 1 1 Ilc 9188
ADD ATM INTERFACE VIRTUAL CHANNEL PVC J1 0 46 1 1 Ilc 9188
ADD ATM INTERFACE VIRTUAL CHANNEL PVC J1 0 47 1 1 Ilc 9188
ADD ATM INTERFACE VIRTUAL CHANNEL PVC J1 0 100 1 1 Ilc 9188
SET VIRTUAL PORT PHYSICAL PORT V4.1 J1
SET VIRTUAL PORT PHYSICAL PORT V4.2 J1
SET VIRTUAL PORT PHYSICAL PORT V4.3 J1
ADD ATM PPP PVC V4.1 0 45
ADD ATM PPP PVC V4.2 0 46
ADD ATM PPP PVC V4.3 0 47
ADD ATM CIP_PVC PVC V7.1 0 100
```

```
CONTEXT ISP1
ADD IP NETWORK ENTRY 192.168.221.40 255.255.255.0 V7.1
SET IP NETWORK MTU 192.168.221.40 9180
ADD IP ROUTE ENTRY 192.168.214.0 255.255.255.0 192.168.221.60 1
ADD RADIUS AUTHENTICATION SERVER ENTRY 1 192.168.214.1 "edge" 5 3 1645
ADD RADIUS ACCOUNTING SERVER ENTRY 1 192.168.214.1 "edge" 5 3 1646
RESET
```

These configurations of an access server are often a complex and time-consuming experience. Suppose that the administrator has to write each command line of this configuration manually, specially if it concerns supporting 2000 users. The risk of excluding commands or creating other incorrect configurations details are not uncommon.

As it was described in chapter 4, the user should only has knowledge about how the network looks like and then fill all necessary data into the panels.

The program generates all command lines in the specific order that the router can handle, in order to establish a connection with the rest of the networks.

The configuration above describes a scenario for PPP over ATM and shows that three users are connected to an ATM card (named J1) plugged into the router via ADSL modem. The router is configured so that each user is connected to a virtual port (V4.1, V4.2 and V4.3) with specified Virtual Path (VP) and Virtual Channel (VC) values. In our case, the user number 1 is connected to the virtual port V4.1 with VP/VC value 0/45, and the user number 2 is connected to the virtual port V4.2 with VP/VC value 0/46. If there are several users connected the number of virtual ports and VP/VC values will increase by one digit.

(V4.x, VP/VC value 0/4x there x = 1, 2, 3...)

From the outgoing side, the ISP1 is connected to the virtual port V7.1 with VP/VC value 0/100.

According to the IP-Routing technology, each connected point must have a specific IP address that must be added to the IP Route Table, in order to establish connection between connected points.

The ISP1 with gateway 192.168.221.60 is connected to the router with IP address 192.168.221.40 and the mask 255.255.255.0.

RADIUS proxy server has an IP address 192 168.214.1 and it is used for authentication of all users.

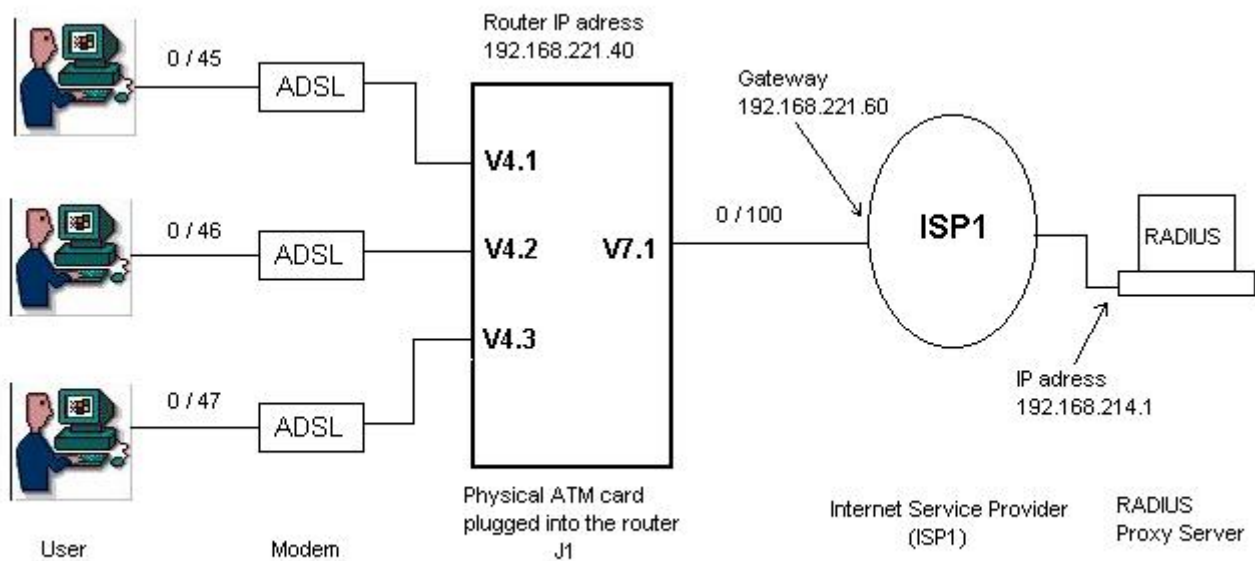


Figure 5.1 PPP over ATM scenario, three users are connected to ISP1 via router

## 5.1 Introduction to Command Usage

This section explains the significance and the functions you can perform with the command lines. These descriptions explain the function of each command, give the format for entering the command, and define parameters and options.

With the configuration script commands, the router can be configured by downloading and executing the file containing a list of configuration commands.

### 5.1.1 Access Partition

In the router, access partition is used to connect the users to the ISP or corporate networks via a secure *Virtual Private Networking (VPN)*. It means that the access partition offers a secure connectivity and keeps customer's traffic separate from the others. The commands associated with access partition will describe as below:

- ❖ The commands, which create or delete an access partition are:

ADD ACCESS PARTITION ENTRY *partition\_name*

Or

DELETE ACCESS PARTITION ENTRY *partition\_name*

Where:

*partition\_name* is the name of ISP or corporate network. For example:

ADD ACCESS PARTITION ENTRY "ISP1"



- ❖ The command which specifies IP addresses for the gateway used to access to the remote ISP or corporate network is:

```
SET ACCESS PARTITION IP GATEWAY partition_name ip_addr
```

Where:

*ip\_addr* specifies the gateway IP network address used to access to remote the ISP or corporate network. For example:

```
SET ACCESS PARTITION IP GATEWAY "ISP1" 192.168.221.60  
[10]
```

### 5.1.2 Service Profile

Service profiles offer a set of access partitions i.e. an ISP or corporate network to connect into. They are mainly used to connect access partitions on virtual ports for PPP over ATM and PPTP tunneling.

- ❖ The commands used to create or delete service profiles are:  
ADD SERVICE PROFILE ENTRY *profile\_name {service\_type}*  
Or  
DELETE SERVICE PROFILE ENTRY *profile\_name*

Where:

*profile\_name* identifies the name of a service profile, such as "PPTP" when PPTP tunneling is used or "ATMSELECT" for PPP over ATM.

*Service\_type* gives the type of service provided, such as CDNR (Called Number Routing Service) which is a dial-in service used for access partitioning in PPTP tunneling.

For example:

```
ADD SERVICE PROFILE ENTRY "ATMSELECT" CDNR
```

- ❖ The command which associates an access partition name with a configured service profile is:

```
SET SERVICE PROFILE ACCESS PARTITION profile_name  
partition_name
```

Where:

*profile\_name* is the name of service profile.

*partition\_name* is the name of access partition.

For example:

SET SERVICE PROFILE ACCESS PARTITION "ATMSELECT"  
"UID.SELECT"

- ❖ The command which is used to configure a service profile for the specified virtual port is:

SET SERVICE PROFILE PORT PROFILE *port\_id service\_profile*  
Where:

*port\_id* specifies the name of the specified virtual port.

*service\_profile* identifies the profile that is associated with the virtual port.

For example:

SET SERVICE PROFILE PORT PROFILE V4.1 "ATMSELECT"

This means that the service profile "ATMSELECT" is used on virtual port V4.1.

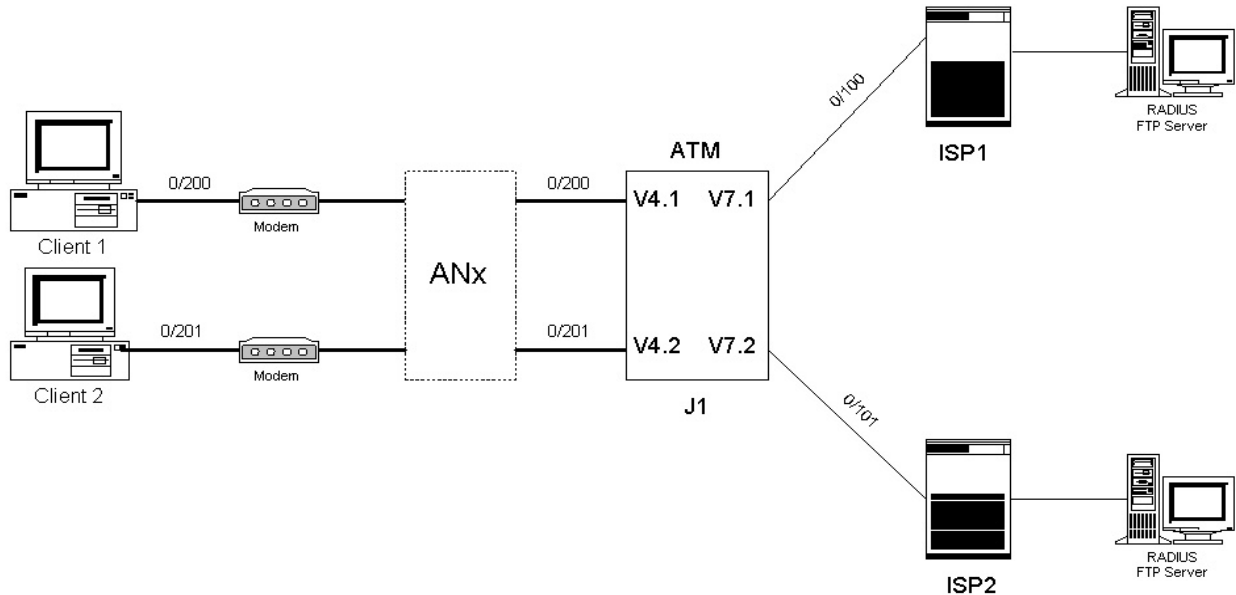
### 5.1.3 Configuring the ATM Interface

This chapter describes some of the commands used in configuring the ATM interface. The following terms is used in this configuration:

- A Virtual Channel Connection (VCC), made up of Virtual Channel Links (VCL) is used as a connection between two endpoints.
- A Virtual Path Connection (VPC) is between two endpoints, made up of Virtual Path Links (VPL).
- PVC (Permanent Virtual Circuit), which is a definition when a VCC or VPC is configured. PVCs are configured between the Network Terminals and the Edge Router, for IP connectivity. PVCs are also used to provide connectivity between the Edge Router and each supported ISP network.
- Virtual Ports are used to connect a virtual link between the ATM port and the network protocols for the interface to be formed, such as IP protocols. The virtual ports are characterized by the selected protocol type and have specific prefixes. Typically V4.x, V6.x, and V7.x are assigned to PPP over ATM, Bridged Ethernet, respectively Classic IP (CIP) over ATM protocol. Where x defines the virtual port number, x = 1 2 3...
- The number of virtual ports depends on the number of configured PVCs.

Each PVC bounds to a virtual port. For example, if we need 10 PVCs, we have to start from virtual port V4.1 up to V4.10. If any of these virtual ports are already occupied, the PVC will be bound to the next virtual port.

The following figure demonstrates the PPP over ATM connection.



**Figure 5.2 Connection of ATM Virtual Ports**

As illustrated in the Figure 5.2, two clients (users) are connected to the ATM card J1. (J1 is the ATM card, which is plugged into the first slot of the router). Each user is connected by one PVC (VPI=0, VCI=200 and VPI=0, VCI=201) to the Virtual Ports V4.1 and V4.2.

From the network side, each ISP will be connected to the one Virtual Port V7.x with different PVCs (VPI=0, VCI=100 and VPI=0, VCI=101). We should notice that we could not connect two ISPs to the same Virtual Port V7.1

- ❖ The commands which create or delete a Permanent Virtual Circuit PVC are as follows:

```
ADD ATM INTERFACE VIRTUAL CHANNEL PVC physical_port vpi vci
rcv_index transmit_index encapsulation
```

Where:

*physical\_port* indicates the ATM port on which to create the PVC.

*vpi* specifies the virtual path identifier. The range of possible values is 0 to 255.

*vci* specifies the virtual channel identifier. The range of possible values is 32 to 25535.

*rcv\_index* and *transmit\_index* is used for receive and transmit direction.

*Encapsulation* specifies the type of encapsulation used on the ATM port and it has the following options: **llc** or **vc\_mux**.

*vc\_mux* is the encapsulation method used for PPP over ATM.

*llc* is an alternative encapsulation method to *vc\_mux* and it is used to carry a number of protocols over a single VC.

An example of configuration for two users will be as below:

```
ADD ATM INTERFACE VIRTUAL CHANNEL PVC J1 0 100 1 1 LLC 9188
ADD ATM INTERFACE VIRTUAL CHANNEL PVC J1 0 101 1 1 LLC 9188
```

These commands show that both users are connected to the same physical ATM port with different VCI.

- ❖ To bind a virtual port to a given physical port we use the command:

```
SET VIRTUAL PORT PHYSICAL PORT virtual_port_id port_id
```

Where:

*virtual\_port\_id* specifies the virtual port.

*port\_id* specifies the physical port.

For example: (also for two users)

```
SET VIRTUAL PORT PHYSICAL PORT V4.1 J1
SET VIRTUAL PORT PHYSICAL PORT V4.2 J1
```

- ❖ The commands used to configure a permanent virtual circuit over the specified PPP virtual port are:

```
ADD ATM PPP PVC virtual_port_id vpi vci
```

Or

```
DELETE ATM PPP PVC virtual_port_id vpi vci
```

An Example of configuration for a PVC over virtual V4.1, configured for PPP, with a VPI=0 and a VCI=200 is:

```
ADD ATM PPP PVC V4.1 0 200
```

- ❖ These commands describes shortly how to set up a PPP over ATM connection:

```
SET SERVICE PROFILE PORT PROFILE V4.1 "ATMSELECT"
```

```
ADD ATM INTERFACE VIRTUAL CHANNEL PVC J1 0 100 1 1 LLC 9188
ADD ATM PPP PVC V4.1 0 200
SET VIRTUAL PORT PHYSICAL PORT V4.1 J1
[11]
```

### 5.1.4 RADIUS Authentication and Accounting

Authentication and accounting are supported in the Edge Router through a RADIUS server. RADIUS authentication begins when the router receives a username and a password from a dial in call.

The following commands are used to configure the authentication server and accounting on the network:

```
ADD RADIUS AUTHENTICATION SERVER ENTRY index ip_address password
retry_interval retry_count UDP_port
```

Where:

*index* indicates the server priority, where servers with lower index numbers are attempted first. The range of valid numbers is 1 to 50.

*ip\_address* is the IP address of the RADIUS authentication server.

*password* is a shared secret that must also be configured on the RADIUS authentication server.

*retry\_interval* is the time between retry attempts to connect to the RADIUS authentication server. The range of possible values is 1 to 60 seconds.

*retry\_count* is the number of times the router attempts to connect to the RADIUS authentication server. The range of possible values is 1 to 10.

The same configuration command is valid for deleting or accounting a RADIUS authentication server. For example:

```
ADD RADIUS AUTHENTICATION SERVER ENTRY 1 192.168.214.1 "edge" 5 3
1645
ADD RADIUS ACCOUNTING SERVER ENTRY 1 192.168.214.1 "edge" 5 3 1645
```

## 6. Structure of Access Setup Wizard ASW

This chapter describes the structure of the program. There are thirteen different classes used in the program to simplify the construction of the code.

This program is developed with JBuilder 3.0 (Borland), based on Sun's Java Development Kit, JDK 2.0 on Microsoft Windows NT 4 Workstation.

The program is divided into the following files (classes):

<i>Class</i>	<i>Description</i>
Handler.java	Controls the whole program
Router.java	Includes the <i>main method</i>
C_Isp.java	Shows the panel with necessary information for creating an ISP
Isp.java	Shows the panel for choosing to create, edit or delete an ISP
IspObj.java	Holds all created data and commands in a vector
PPPoATM.java	Create the PPP over ATM panel
PPTP.java	Create the PPTP panel
Logo.java	Shows the logo for each panel
PopUpError.java	Shows a warning message in case of invalid data input
Txt.java	Shows the text area below the panels
Vc.java	Checks the VC values
Vp.java	Checks the VP values
Welcome.java	Shows the initial panel

Each class has its own functionality and it consists of several *methods*, which is used in different part of the class.

The class, which controls the program, is `Handler`. The `Handler` class contains different *states*, and each *state* takes appropriate action depending on received *event*. The GUI, which is integrated in the program also, depends on the *state* and the actual *event*, which is handling.

For example, if the program is in the *S\_ISP* (State ISP), it will remain there and wait for an *event* to handle. If the *event* is to create an ISP (*Event ISP Create ISP, E\_ISP\_CISP*), then the method `handle_E_ISP_CISP( )` will be called, and it shows the next frame, which will be ISP Settings.

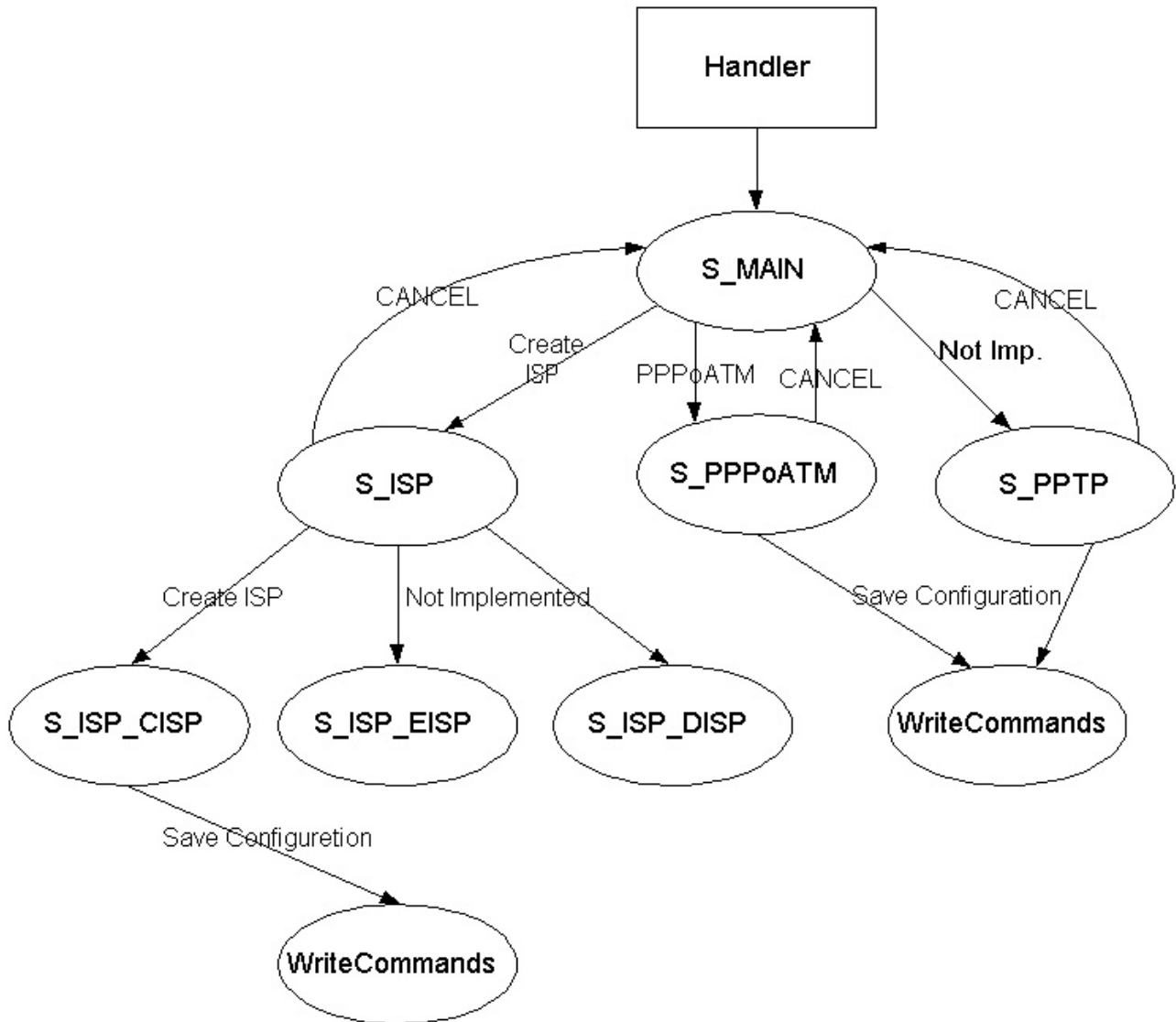
The following *states* and *events* are included the `Handler` class:

<i>States</i>	<i>Description</i>
S_MAIN	The initial state, main panel
S_ISP	State, which shows options for creating, editing or deleting ISP
S_PPPoATM	State for choosing PPP over ATM scenario
S_PPTP	State for choosing PPTP scenario
S_ISP_INIT	Brings the initial panel for creating an ISP
S_ISP_CISP	State for creating an ISP
S_ISP_EISP	State for editing an ISP
S_ISP_DISP	State for deleting an ISP

<i>Event</i>	<i>Description</i>
E_ISP_INIT	Event for initiating the ISP panel
E_ISP_CISP	Event for showing the ISP panel
E_ISP_CANCEL	Exit the ISP panel
E_H_ISP	Event for creating an ISP
E_H_PPPoATM	Event for configuring the PPP over ATM scenario
E_H_PPTP	Event for configuring the PPTP scenario
E_CISP_OK	Accepting the new configuration, press OK button
E_CISP_CANCEL	Deleting the new configuration, press CANCEL button

## 6.1 The Flow Diagram of the Handler class

Diagram 6.1 shows in general, the flow of the Handler class.



**Diagram 6.1 The Flow Diagram of the Handler class**

The State ISP (*S\_ISP*) is divided into three states among others,

- State ISP Create ISP (*S\_ISP\_CISP*)
- State ISP Edit ISP (*S\_ISP\_EISP*)
- State ISP Delete ISP (*S\_ISP\_DISP*).

The last two options are not implemented.



The Router class which includes the *main()* method, initiates the Handler class and calls its *run()* method to run the program.

Each state has its own “handle” method, which updates the state of the Handler, (*S\_MAIN*).

## 6.2 Create ISP

When the program is in the State ISP (*S\_ISP*), it is waiting to receive an event. For activating the State ISP, we are using the method called *handle\_S\_ISP()*.

This method has only two valid events:

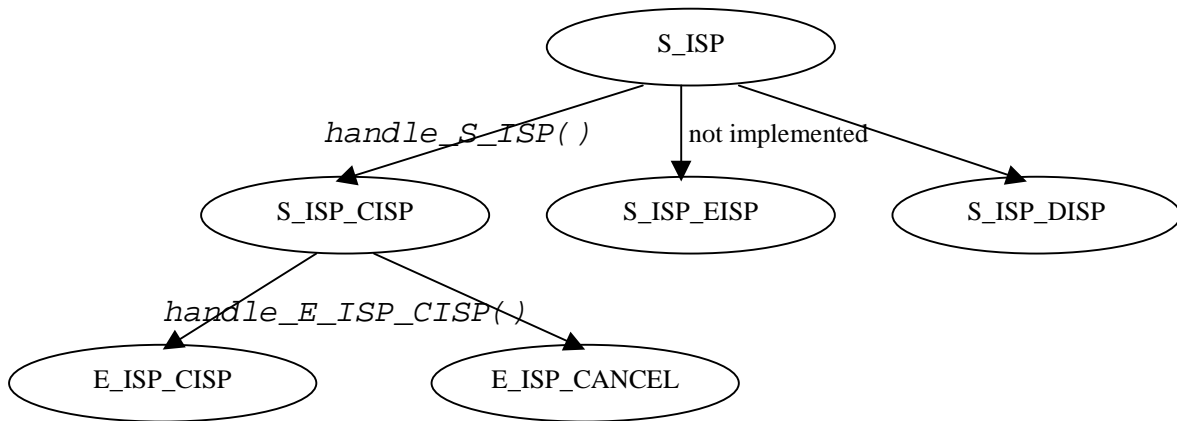
- *E\_ISP\_CISP* for creating an ISP
- *E\_ISP\_CANCEL*, which exits the state and return to *S\_MAIN* state.

In the case of creating an ISP, *E\_ISP\_CISP* (Event ISP Create ISP), it returns to *handle\_E\_ISP\_CISP()* method.

This method brings the frame containing information of ISP settings.

A popUpInformation message is also included in this method to inform the user that she/he has used valid data and configuration for the ISP was successfully done.

Diagram 6.2 illustrates the state *S\_ISP*.



**Diagram 6.2 The Flow Diagram of the State ISP (*S\_ISP*)**

The state *E\_ISP\_CANCEL* is used as a handler event to exit this state and return it back to *S\_MAIN* state. At this step, as it is mentioned before, the state is waiting to receive an event and the same process continues.

### 6.3 PPP over ATM Configuration

Another method used in the `Handler` class is `handle_S_PPPoATM()` method and it contains of two states:

- `E_PPPoATM_OK`
- `E_PPPoATM_CANCEL`

This method is also waiting for receiving an event. When it gets an event, it brings the appropriate frame, and shows it to the user to make a configuration of PPP over ATM scenario.

It is also included `popUpInformation` message frame to inform the user about configuration status or invalid data.

The event, which exits this state, is `E_PPPoATM_CANCEL`. It removes the actual panel from the frame and returns to the `S_MAIN` state.

This scenario is illustrated in Diagram 6.3.

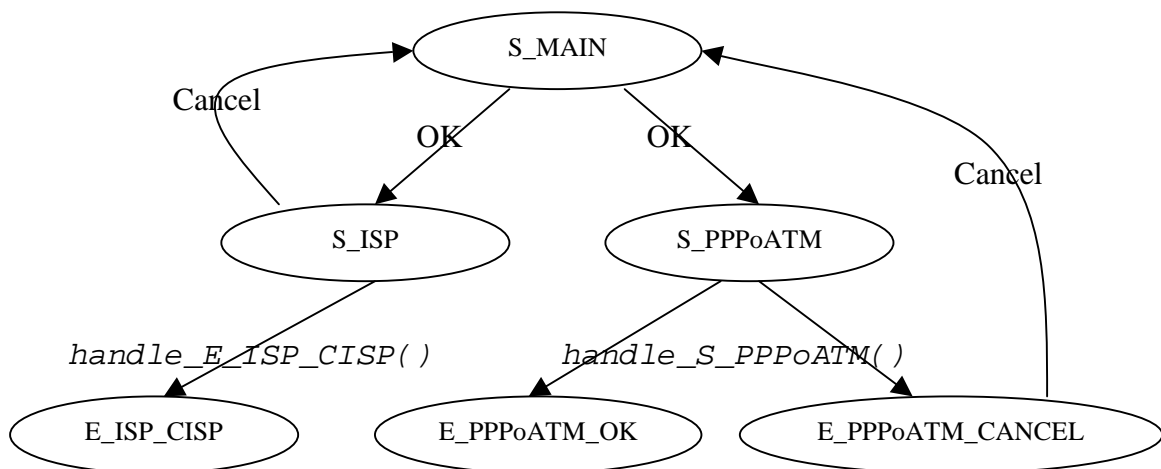


Diagram 6.3 The Flow Diagram of the State `S_PPPoATM`

## 7. Conclusions

Access Setup Wizard (ASW) is a Java-based configuration tool, and the Graphical User Interface (GUI) Framework used for configuration different network scenarios in the Ericsson's Broadband Access Server (BAS).

The configuration of an Access Server is often a complex task. The administrator should know exactly how to configure the BAS, and also how the network is established. Information about ATM PVCs, IP addressing is also needed.

Suppose that in a network scenario, there are many users with different platforms (PC, MAC and UNIX) that are connected.

In order to obtain an easy way to make a configuration, and offer a secure connectivity to an ISP or a corporate network, access to a simple tool is necessary. Developing a Java-based program would be a solution to achieve this tool, which is required. Java is a most powerful programming language for Internet- and network consistency, because Java is suppose to be platform independent and it will be accessible on any machine.

To develop such a program was not an easy task, and there were many factors that I had to deal with.

Before doing any implementation, I had to gather a lot of knowledge about the architecture of the network and those components, which are used in such a network. I was also studying about the ATM technology and the Edge Router functionality in general.

I put myself in a position as an administrator, and I tried to understand what information or knowledge I needed in order to find out how the program should be developed.

The second step was to improve my knowledge about Java programming language. Using on line information about Java, was a great help.

During this work I got more knowledge about networking, TCP/IP, ATM technology and also functionality of different network scenarios, such as PPP over ATM, Point to Point Tunneling Protocol (PPTP), and Network Address Translation (NAT). I have also got more experience to work individually and in a team.

The feature of the Access Setup Wizard (ASW) is a way to give the administrator more possibility to add or extend this program for configuration of other network scenarios like NAT and PPP over Ethernet.

Different functions can be implemented, in order to be able to make the script file, even for other scenarios.

## 8. References

1. Summers, C. 1999. *ADSL: standards, implementation, and architecture*. Boca Raton, Fla. : CRC Press. 0-8493-9595-X
2. Kumar, B. 1995. *Broadband communications : professional's guide to ATM, frame relay, SMDS, SONET and B-ISDN*. New York : McGraw-Hill. 0-07-035968-7
3. Händel, R. 1998. *ATM networks : concepts, protocols, applications*. Harlow : Addison-Wesley. 0-201-17817-6
4. Cuthbert, L.G. 1993. *ATM : the broadband telecommunications solution*. London : Institution of Electrical Engineers. 0-85296-815-9
5. McDysan, D.E. 1995. *ATM : theory and applications*. New York : McGraw-Hill. 0-07-060362-6
6. Stevens, W.R. 1994. *TCP/IP illustrated, Vol. 1 : The protocols*. Reading, Mass. : Addison-Wesley. 0-201-63346-9
7. Horstmann, C. S. 1999. *Core Java 2, Vol. 1 : Fundamentals*. Upper Saddle River, NJ. : Prentice Hall. 0-13-081933-6
8. Skansholm, J. 2000. *Java direkt*. Lund : Studentlitteratur. 91-44-01244-6
9. Ek, J. 1996. *Java-programmering : en introduktion : lär dig programmera Java för Internet*. Stockholm : Helsingfors : Pagina. 91-636-0419-1
  
10. AXC 706 Command Reference guide 1.0 Ericsson Documentation
11. AXI 510 Edge Router, User Guide V1.1 Ericsson Documentation
12. AXI 510 Edge Router, Overview Course Ericsson Documentation

## 9. Additional Information Sources

### Online Documentation

#### Network Solutions:

[CISCO] Cisco's services webpage,

URL: <http://www.cisco.com/warp/public/779/servpro/solutions/>  
<http://www.cisco.com/univercd/cc/td/doc/product/atm/>.

[ASCEND] Ascend's VPN solutions webpage,

URL: <http://www.ascend.com/3543.html>.

[BAY] Bay Networks Virtual Private Network solutions webpage,

URL: <http://www.baynetworks.com/solutions/vpn/>.

[3COM] 3Com's ATM solutions webpage,

URL: <http://www.3com.com/nsc/500374.html>.

[LUCENT] Lucent's solutions webpage,

URL: <http://www.lucent.com/serviceprovider/solutions/>.

[IETF] Internet Engineering Task Force homepage,

URL: <http://www.ietf.org>.

#### Java Documentation

[SUN] **Java Tutorial**

URL: <http://developer.java.sun.com/developer/onlineTraining/>

URL: <http://java.sun.com/docs/books/tutorial/uiswing/>

URL: <http://developer.java.sun.com/developer/onlineTraining/GUI/Swing2/>

URL: <http://java.sun.com/products/jlf/dg/index.htm>

#### API Documentation

URL: <http://java.sun.com/products/jdk/1.2/docs/api/>

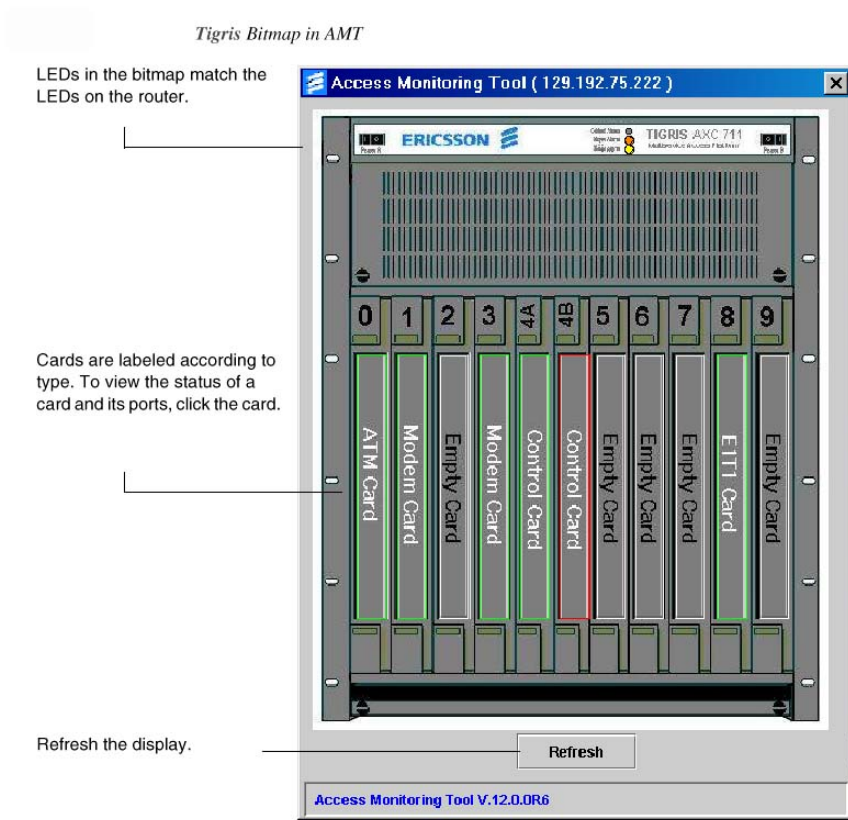
[Progsources] URL: <http://www.progsources.com/java.html>

[Earthweb] URL: <http://gamelan.earthweb.com/dlink.index-jhtml.72.1082.-.-.jhtml>

# Appendix A

## Ericsson's AXC 706 Broadband Access Server

Ericsson has developed AXC 706 Broadband Access Server in two different variants. One with six slots and the other with eleven slots, reserved for plugging the ATM and Ethernet cards, as it shows in the picture below.



## Appendix B

### The Handler class

```
package project;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.UIManager;
import javax.swing.border.*;
import java.lang.Thread;
import java.util.*;
import java.net.*;
import java.io.*;
import java.awt.TextArea.*;

public class Handler {
    public static final int S_MAIN = 0;
    public static final int S_ISP = 1;
    public static final int S_PPPoATM = 2;
    public static final int S_PPTP = 3;
    public static final int E_H_NO_EVENT = 0;
    public static int event = E_H_NO_EVENT;
    public static final int E_H_ISP = 1;
    public static final int E_H_PPPoATM = 2;
    public static final int E_H_PPTP = 3;
    public static final int E_H_NAT = 4;
    public static final int E_H_DISP_F = 5;
    public static final int E_H_SAVE_F = 6;

    public static final int S_ISP_INIT = 0;
    public static final int S_ISP_CISP = 1;
    public static final int S_ISP_EISP = 2;
    public static final int S_ISP_DISP = 3;

    public static final int E_ISP_NO_EVENT = 0;
    public static final int E_ISP_CISP = 1;
    public static final int E_ISP_EISP = 2;
    public static final int E_ISP_DISP = 3;
    public static final int E_ISP_CANCEL = 4;

    public static final int E_CISP_NO_EVENT = 0;
    public static final int E_CISP_CANCEL = 1;
    public static final int E_CISP_OK = 2;

    public static final int E_PPPoATM_NO_EVENT = 0;
    public static final int E_PPPoATM_CANCEL = 1;
    public static final int E_PPPoATM_OK = 2;

    public static final int E_PPTP_NO_EVENT = 0;
    public static final int E_PPTP_CANCEL = 1;
    public static final int E_PPTP_OK = 2;
    private static final int MAX_NUM_VP = 256;
```

```

private static final int MAX_NUM_VC = 65536;
public static final int NUM_OF_PHY_PORTS = 4;
JButton s1b, s2b, s3b, dcb, scb, exit;

public IspConfig ic;
public Rectangle extR, logoR, handR, scenR, textR;
public Container cp;

Vector cIsps = new Vector();
Vector cmd = new Vector();
int ispNum = 1;
int numOfIsps = 0;
Vector[] pvc = new Vector[NUM_OF_PHY_PORTS];
public int state;
public boolean ISP_PR, EXIT_PR;
    public Txt txt;
    public Isp isp;
    public C_Isp c_isp;
    public PPPoATM pppoatm = null;
    public PPTP pptp = null;
    public HandlerFrame hf;
    HandlerPanel hp;
    Welcome wel;
    PopUpError p = new PopUpError();

public Handler () {
    initShow();
    // Initialize vectors that will hold the VPs for each physical port
    for(int i = 0; i < Values.NUM_OF_PHY_PORTS; i++) {
        pvc[i] = new Vector();
        pvc[i].trimToSize();
    }
    // Initialize state
    state = S_MAIN;
}

public int handle_S_MAIN() {
    int event;
    if (numOfIsps > 0) {
        s1b.setEnabled(true); //PPPoATM Button
        s2b.setEnabled(true); //PPTP Button
        s3b.setEnabled(false); //NAT Button
        dcb.setEnabled(true); //Display Button
        scb.setEnabled(true); //Save Button
    }
    else {
        scb.setEnabled(false);
        s1b.setEnabled(false);
        s2b.setEnabled(false);
        s3b.setEnabled(false);
    }

    cp.add(wel);
    hf.setVisible(false);
    hf.setVisible(true);
    event = waitForEvent();
    cp.remove(wel);
    hf.setVisible(false);
}

```



```

switch(event) {
    case E_H_ISP:

        // Update the content pane of the main frame
        // Update what is displayed
        isp = new Isp(scenR);
        cp.add(isp.ip);

        hp.ispb.setEnabled(false);
        state = S_ISP; // get the new state, update state
        break;

    case E_H_PPPOATM:
        pppoatm = new PPPoATM(scenR, pvc);
        cp.add(pppoatm.pppoatmp);
        hf.setVisible(true);
        state = S_PPPOATM;
        break;

        case E_H_PPTP:
            pptp = new PPTP(scenR, pvc);
            cp.add(pptp.pptpp);
            hf.setVisible(true);
            state = S_PPTP;
            break;

}
    hf.setVisible(true);
return state;
}

#####

public int handle_S_PPPOATM() {
    int event;
    event = pppoatm.waitForEvent();
    switch(event) {
        case E_PPPOATM_OK:
            if (pppoatm.occPvcFound) {
                txt.t.append(pppoatm.str);
                p.popUpErrorMsg("The PVCs shown in the text area are occupied. Please try again.");
                txt = new Txt(textR); cp.add(txt);
                cp.remove(pppoatm.pppoatmp);
                pppoatm = new PPPoATM(scenR, pvc);
                cp.add(pppoatm.pppoatmp);
                hf.setVisible(true);
                return S_PPPOATM;
            }
            else {
                cp.remove(pppoatm.pppoatmp);
                cp.remove(txt); txt = new Txt(textR); cp.add(txt);
                hf.setVisible(true);
                hp.ispb.setEnabled(true);
                p.popUpInformationMsg("Your configuration has successfully done." + "\n" +
                    "You may display configuration or save it to a file.");
            }
            break;

        case E_PPPOATM_CANCEL:

```

```

        cp.remove(pppoatm.pppoatmp);
        cp.remove(txt); txt = new Txt(textR); cp.add(txt);
        break;
    }
    hf.setVisible(true);
    return S_MAIN;
}

#####

public void handle_E_ISP_CISP() {          // ISP Settings
    int event;
    hf.setVisible(false);
    cp.remove(isp.ip);
    c_isp = new C_Isp(scenR, ispNum, pvc);
    cp.add(c_isp.cip);
    hf.setVisible(true);
    event = c_isp.waitForEvent();
    switch(event) {
        case E_CISP_OK:
            cIsps.add(c_isp.ispObj);
            numOfIsps++; ispNum++;
            cp.remove(c_isp.cip);
            hp.ispb.setEnabled(true);

            p.popUpInformationMsg("Your configuration has successfully done." + "\n" +
                "You may configure another ISP or select an Access Method.");

            break;
        case E_CISP_CANCEL:
            cp.remove(c_isp.cip);
            hp.ispb.setEnabled(true);
            break;
    }
}
#####

public int handle_S_ISP() {              ///Create, Edit, Delete ISP
    boolean flag1 = false;
    int event;
    switch(isp.state) {
        case S_ISP_INIT:
            event = isp.waitForEvent();
            switch(event) {
                case E_ISP_CISP:
                    handle_E_ISP_CISP();

// NOT Implementet//
                /*
                case E_ISP_EISP:
                    handle_E_ISP_EISP();
                    break;
                case E_ISP_DISP:
                    handle_E_ISP_DISP();
                    break;
                */
                case E_ISP_CANCEL:
                    cp.remove(isp.ip);
                    hp.ispb.setEnabled(true);
            }
    }
}

```

```

        break;
    } // end of switch

    return S_MAIN;
}

#####

public void run() {

    while(true) {
        switch(state) {

            case S_MAIN: // waiting for create ISP button to be pressed
                state = handle_S_MAIN();
                break;

            case S_ISP:
                state = handle_S_ISP();
                break;

            case S_PPPOATM:
                state = handle_S_PPPOATM();
                break;

            // NOT Implementet//
            /*
            case S_PPTP:
                state = handle_S_PPPTP();
                break;

            case S_NAT:
                state = handle_S_NAT();
                break;
            */

        }

        try { Thread.sleep(100); } catch (InterruptedException e) {}

    }

}

#####
// For writing commands//

public void writeCmds() {
    try {

        IspObj io;
        cIspS.trimToSize();
        FileWriter fw = new FileWriter("C:\\ScriptFile\\script.txt");
        PrintWriter pw = new PrintWriter(fw);

        // put 2 first lines

        String asw = "TIGRIS";
        pw.println("SET SCRIPT VERSION (Access Setup Wizard Version 5.1)");
        pw.println(("SET PROMPT" + " " + "\"" + asw + "\""));
        pw.println("SET VIRTUAL PORT COUNT 10 0 0 0 " + pppoatm.numOfUsers +
            " " + "0" + " " + "0" + " " + numOfIspS );
    }
}

```

```

for (int i = 0; i < cIsps.capacity(); i++) {
    io = (IspObj) cIsps.elementAt(i);
    pw.println( io.cmd.elementAt(io.counter++) );
    pw.println( io.cmd.elementAt(io.counter++) );
}

pw.println( "SET RADIUS PORT COUNT 50");

for (int i = 0; i < cIsps.capacity(); i++) {
    io = (IspObj) cIsps.elementAt(i);
    for (int j = 0; j < 3; j++)
        pw.println( io.cmd.elementAt(io.counter++) );
}

for (int i = 0; i < 3; i++)
    pw.println( pppoatm.vcmd.elementAt(pppoatm.counter++) );

for (int i = 0; i < cIsps.capacity(); i++) {
    io = (IspObj) cIsps.elementAt(i);
    pw.println( io.cmd.elementAt(io.counter++) );
}

for(int i= 0; i < pppoatm.numOfUsers; i++) {
    pw.println( pppoatm.vcmd.elementAt(pppoatm.counter++) );
}

pw.println( "ADD ATM TRAFFIC DESCRIPTOR 1 UBR 14800000");

for(int i= 0; i < pppoatm.numOfUsers; i++) {
    pw.println( pppoatm.vcmd.elementAt(pppoatm.counter++) );
}

for (int i = 0; i < cIsps.capacity(); i++) {
    io = (IspObj) cIsps.elementAt(i);
    pw.println( io.cmd.elementAt(io.counter++) );
}

int cap = pppoatm.vcmd.capacity() - pppoatm.counter;
for (int i = 0; i < cap; i++)
    pw.println( pppoatm.vcmd.elementAt(pppoatm.counter++) );

for (int i = 0; i < cIsps.capacity(); i++) {
    io = (IspObj) cIsps.elementAt(i);
    pw.println( io.cmd.elementAt(io.counter++) );
}

for (int i = 0; i < cIsps.capacity(); i++) {
    io = (IspObj) cIsps.elementAt(i);
    for (int j = io.counter; j < io.cmd.capacity(); j++)
        pw.println( io.cmd.elementAt(io.counter++) );
}

pw.println("RESET");

fw.close();
for (int i = 0; i < cIsps.capacity(); i++) {
    io = (IspObj) cIsps.elementAt(i);
    io.counter = 0;
}

```

```

        pppoatm.counter = 0;
    }
    catch (Exception e) {}
}

#####

// Write the commands to the Textarea //

public void copyToTextArea() {
    try {
        IspObj io;
        cIsps.trimToSize();
        String asw = "ASW_TIGRIS";
        txt.t.append( (String) ("SET SCRIPT VERSION (Access Setup Wizard Version 5.1)")+"\n");
        txt.t.append( (String) ("SET PROMPT" + " " + "\""+asw+"\"")+"\n");
        txt.t.append( (String) ("SET VIRTUAL PORT COUNT 10 0 0 0 " +
            + pppoatm.numOfUsers + " " + "0" + " " + "0" +
            " " + numOfIsps )+"\n");

        for (int i = 0; i < cIsps.capacity(); i++) {
            io = (IspObj) cIsps.elementAt(i);
            txt.t.append( (String) ( io.cmd.elementAt(io.counter++) )+"\n");
            txt.t.append( (String) ( io.cmd.elementAt(io.counter++) )+"\n");
        }

        txt.t.append( (String) ("SET RADIUS PORT COUNT 50")+"\n");

        for (int i = 0; i < cIsps.capacity(); i++) {
            io = (IspObj) cIsps.elementAt(i);
            for (int j = 0; j < 3; j++)
                txt.t.append( (String) ( io.cmd.elementAt(io.counter++) )+"\n");
        }

        for (int i = 0; i < 3; i++)
            txt.t.append( (String) ( pppoatm.vcmd.elementAt(pppoatm.counter++) )+"\n");

        for (int i = 0; i < cIsps.capacity(); i++) {
            io = (IspObj) cIsps.elementAt(i);
            txt.t.append( (String) ( io.cmd.elementAt(io.counter++) )+"\n");
        }

        for (int i = 0; i < pppoatm.numOfUsers; i++) {
            txt.t.append( (String) ( pppoatm.vcmd.elementAt(pppoatm.counter++) )+"\n");
        }

        txt.t.append( (String) ("ADD ATM UBR TRAFFIC DESCRIPTOR 1 14800000")+"\n");

        for (int i = 0; i < pppoatm.numOfUsers; i++) {
            txt.t.append( (String) ( pppoatm.vcmd.elementAt(pppoatm.counter++) )+"\n");
        }

        for (int i = 0; i < cIsps.capacity(); i++) {
            io = (IspObj) cIsps.elementAt(i);
            txt.t.append( (String) ( io.cmd.elementAt(io.counter++) )+"\n");
        }

        int cap = pppoatm.vcmd.capacity() - pppoatm.counter;
        for (int i = 0; i < cap; i++)

```

```

        txt.t.append( (String) ( pppoatm.vcmd.elementAt(pppoatm.counter++) )+"\n");

for (int i = 0; i < cIsps.capacity(); i++) {
    io = (IspObj) cIsps.elementAt(i);
    txt.t.append( (String) ( io.cmd.elementAt(io.counter++) )+"\n");
}

for (int i = 0; i < cIsps.capacity(); i++) {
    io = (IspObj) cIsps.elementAt(i);
    for (int j = io.counter; j < io.cmd.capacity(); j++)
        txt.t.append( (String) ( io.cmd.elementAt(io.counter++) )+"\n");
}

txt.t.append( (String) ("RESET")+"\n");

for (int i = 0; i < cIsps.capacity(); i++) {
    io = (IspObj) cIsps.elementAt(i);
    io.counter = 0;
}
pppoatm.counter = 0;
}
catch (Exception e) {}
}

#####

public int waitForEvent() {
    int curEvent;
    while (event == E_H_NO_EVENT)
        try { Thread.sleep(100); } catch (InterruptedException e) {}
    curEvent = event;
    event = E_H_NO_EVENT;
    return curEvent;
}

void initShow() {
    hf = new HandlerFrame();
    hp = new HandlerPanel(handR);
    Logo lg = new Logo(logoR);
    wel = new Welcome(scenR);
    txt = new Txt(textR);
    cp.add(lg); cp.add(hp); cp.add(txt);
    hf.setVisible(true);
}

// HandlerFrame
public class HandlerFrame extends JFrame {

    double LOGO_P_MUL_W = 1;
    double LOGO_P_MUL_H = 1;
    double HANDLER_P_MUL_W = 0.3;
    double HANDLER_P_MUL_H = 0.7;
    double SCEN_P_MUL_W = 1 - HANDLER_P_MUL_W;
    double SCEN_P_MUL_H = 0.7;
    double TEXT_P_MUL_W = 1;
    double TEXT_P_MUL_H = 1 - SCEN_P_MUL_H;// - LOGO_P_MUL_H;

    double[] extFrame_mul = {0.9, 0.9};
    double[] logoFrame_mul = {LOGO_P_MUL_W, LOGO_P_MUL_H};
    double[] handFrame_mul = {HANDLER_P_MUL_W, HANDLER_P_MUL_H};
}

```

```

double[] scenFrame_mul = {SCEN_P_MUL_W, SCEN_P_MUL_H};
double[] textFrame_mul = {TEXT_P_MUL_W, TEXT_P_MUL_H};

int extX, extY, logoX, logoY, handX, handY, scenX, scenY, textX, textY,
    extW, extH, logoW, logoH, handW, handH, scenW, scenH, textW, textH;
int w, h;

public HandlerFrame() {

    Toolkit tk = Toolkit.getDefaultToolkit();
    Dimension d = tk.getScreenSize();
    w = d.width; h = d.height;

    extW = (int) (w * extFrame_mul[0]);
    extH = (int) (h * extFrame_mul[1]);
    extX = w/2 - extW/2;
    extY = h/2 - extH/2;

    logoX = 0;
    logoY = 0;
    logoW = extW;
    logoH = 50;
    handX = 0;
    handY = logoH;
    handW = (int) (extW * handFrame_mul[0]);
    handH = (int) (extH * handFrame_mul[1]);

    scenX = handW;
    scenY = logoH;
    scenW = (int) (extW * scenFrame_mul[0]);
    scenH = (int) (extH * scenFrame_mul[1]);

    textX = 0;
    textY = logoH + handH;
    textW = (int) (extW * textFrame_mul[0]);
    textH = (int) (extH * textFrame_mul[1]) - 50;

    extR = new Rectangle(extX, extY, extW, extH);
    logoR = new Rectangle(logoX, logoY, logoW, logoH);
    handR = new Rectangle(handX, handY, handW, handH);
    scenR = new Rectangle(scenX, scenY, scenW, scenH);
    textR = new Rectangle(textX, textY, textW, textH);

    setBounds(extR);
    setResizable(false);
    setTitle("Access Setup Wizard");
    addWindowListener(
        new WindowAdapter() {
            public void windowClosing(WindowEvent e) { System.exit(0); }
        }
    );
    cp = getContentPane();
    cp.setLayout(null);
}
} // end of HandlerFrame

// HandlerPanel
class HandlerPanel extends JPanel implements ActionListener {

    public boolean E_ISP;

```

```

public JButton ispb;

public HandlerPanel(Rectangle R) { //Constructor//
    int w = R.width, h = R.height;
    int lfsz, bfsz;
    int dis1, dis2, dis3;
    dis1 = h/40;
    dis2 = 2 * dis1;
    dis3 = 4 * dis1;

    Border border1, border2;
    setBounds(R);

    lfsz = (int) h/24;
    bfsz = (int) h/34;
    border1 = BorderFactory.createCompoundBorder(BorderFactory.createEtchedBorder
        (Color.black,new java.awt.Color(134, 134, 134)),
        BorderFactory.createEmptyBorder(10,10,10,10));
    setBorder(BorderFactory.createLineBorder(Color.black));

    border2 = BorderFactory.createCompoundBorder(BorderFactory.createBevelBorder
        (BevelBorder.RAISED,Color.white,Color.white,
        new java.awt.Color(134, 134, 134),
        new java.awt.Color
        (93, 93, 93)),BorderFactory.createEmptyBorder(5,5,5,5));

    Box b = Box.createVerticalBox();

    JLabel ispConf = new JLabel("ISP Configuration", JLabel.CENTER);
    ispConf.setFont( new Font("Dialog", Font.BOLD, lfsz) );
    ispConf.setForeground(Color.darkGray);
    b.add(Box.createVerticalStrut(dis1/3)); b.add(ispConf);

    ispb = new JButton("Configure ISP");
    ispb.setBackground(new Color(192, 192, 255));
    ispb.addActionListener(this);
    ispb.setFont(new Font ("Dialog", Font.BOLD, bfsz));
    b.add(Box.createVerticalStrut(dis1)); ispb.setBorder(border2);
    b.add(ispb);

    JLabel access = new JLabel("Access Method", JLabel.CENTER);
    access.setFont( new Font("Dialog", Font.BOLD, lfsz) );
    access.setForeground(Color.darkGray);
    b.add(Box.createVerticalStrut(dis2)); b.add(access);

    s1b = new JButton("PPP over ATM");
    s1b.setBackground(new Color(192, 192, 255));
    s1b.addActionListener(this);
    s1b.setFont(new Font ("Dialog", Font.BOLD, bfsz));
    s1b.setBorder(border2);
    s1b.setEnabled(false);
    b.add(Box.createVerticalStrut(dis1)); b.add(s1b);

    s2b = new JButton("PPTP");
    s2b.setBackground(new Color(192, 192, 255));
    s2b.addActionListener(this);
    s2b.setFont(new Font ("Dialog", Font.BOLD, bfsz));
    b.add(Box.createVerticalStrut(dis1)); s2b.setBorder(border2);
    s2b.setEnabled(false);
    b.add(s2b);

```



```

s3b = new JButton("NAT");
s3b.setBackground(new Color(192, 192, 255));
s3b.addActionListener(this);
s3b.setFont(new Font ("Dialog", Font.BOLD, bsize));
b.add(Box.createVerticalStrut(dis1)); s3b.setBorder(border2);
s3b.setEnabled(false);
b.add(s3b);

JLabel conf = new JLabel("Save Configuration", JLabel.CENTER);
conf.setFont( new Font("Dialog", Font.BOLD, lsize) );
conf.setForeground(Color.darkGray);
b.add(Box.createVerticalStrut(dis2)); b.add(conf);

dcb = new JButton("Display configuration");
dcb.setBackground(new Color(192, 192, 255));
dcb.addActionListener(this);
dcb.setFont(new Font ("Dialog", Font.BOLD, bsize));
b.add(Box.createVerticalStrut(dis1)); dcb.setBorder(border2);
dcb.setEnabled(false);
b.add(dcb);

scb = new JButton("Save configuration");
scb.setBackground(new Color(192, 192, 255));
scb.addActionListener(this);
scb.setFont(new Font ("Dialog", Font.BOLD, bsize));
b.add(Box.createVerticalStrut(dis1)); scb.setBorder(border2);
scb.setEnabled(false);
b.add(scb);

JLabel exitL = new JLabel("Exit Configuration", JLabel.CENTER);
exitL.setFont( new Font("Dialog", Font.BOLD, lsize) );
exitL.setForeground(Color.darkGray);
b.add(Box.createVerticalStrut(dis2)); b.add(exitL);

exit = new JButton("Exit");
exit.setBackground(new Color(192, 192, 255));
exit.addActionListener(this);
exit.setFont(new Font ("Dialog", Font.BOLD, bsize));
b.add(Box.createVerticalStrut(dis1)); exit.setBorder(border2);
b.add(exit);
add(b);
}

public void actionPerformed(ActionEvent evt) {
    Object source = evt.getSource();
    if (source == exit)
        System.exit(0);
    if (source == ispb)
        event = E_H_ISP;
    else if (source == s1b)
        event = E_H_PPPoATM;

    else if (source == s2b)
        event = E_H_PPTP;

    else if (source == scb)
        writeCmds();
    else if (source == dcb)

```

```
        copyToTextArea();
    } // end of actionPerformed
} // end of panel //
} // end of Handler
```

## The C\_Isp class

```
package project;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.UIManager;
import javax.swing.border.*;
import java.lang.Thread;
import java.util.*;
import java.net.*;

public class C_Isp {
    private int MAX_NUM_VP2 = 255;
    private int MIN_NUM_VP2 = 0;
    private int MIN_NUM_VC2 = 32;

    public static final int E_NO_EVENT = 0;
    public static final int E_CANCEL = 1;
    public static final int E_OK = 2;
    public int event = E_NO_EVENT;
    boolean ERROR_FOUND = false;
    String ISPConf[] = new String[30];
    public IspObj ispObj;
    public C_IspPanel cip;
    PopUpError p = new PopUpError();
    Vector[] pvc;
    JButton cisp, eisp, disp;
    Vector comboStrs = new Vector();
    JButton cancel, ok;
    JTextField nameField, gateField, ipField, maskField, vpField, vcField, radIPaddrField, edgeField;
    String tmpStr1, name, ispIPaddr, rtIPaddr, rtMask, vpStr, vcStr, radIPaddr, str8;
    String secret1 = "edge";
    JComboBox portList;
    int vpI, vcI, ispNum;
    Vector vcmd = new Vector();
    public int waitForEvent() {
        int curEvent;
        while (event == E_NO_EVENT)
            try { Thread.sleep(100); } catch (InterruptedException e) {}
        curEvent = event;
        event = E_NO_EVENT;
        return curEvent;
    }

    public C_Isp(Rectangle R, int ispNum, Vector[] pvc) { // Constructor of C_Isp
        cip = new C_IspPanel(R);
        this.ispNum = ispNum;
        this.pvc = pvc;
    }

    class C_IspPanel extends JPanel implements ActionListener {

        JLabel setting;
        JButton CreateIsp, edit, DeleteIsp, cancel;
        private IspConfig ispFrame ;
```

```

String str = "ISP1";
String selectPort;

public C_IspPanel(Rectangle R) { // Constructor of C_IspPanel
    Font f, f2;
    int w, h, lh, remh, lfsz, bfsz, dis1, dis2, dis3, bsize, col;
    int space1, space2, space3;
    w = R.width;
    h = R.height;

    dis1 = h/40;
    dis2 = w/20;
    dis3 = 4 * dis1;

    w = R.width;
    h = R.height;
    lh = (int) (h * 0.1);
    remh = h - lh;
    lfsz = (int) (h/24);
    f = new Font("Dialog", Font.BOLD, lfsz);
    f2 = new Font("Dialog", Font.PLAIN, lfsz);
    bfsz = (int) (h/34);
    col = (int) (w/88);
    setBounds(R);
    setLayout(null);
    Border border1, border2;
    border1 = BorderFactory.createCompoundBorder(BorderFactory.createEtchedBorder
        (Color.white,new java.awt.Color(134, 134, 134)),
        BorderFactory.createEmptyBorder(5,5,5,5));

    border2 = BorderFactory.createCompoundBorder(BorderFactory.createBevelBorder
        (BevelBorder.RAISED,Color.white,Color.white,new java.awt.Color(134, 134, 134),
        new java.awt.Color(93, 93, 93)),BorderFactory.createEmptyBorder(5,5,5,5));

    JPanel p1, p2, p3, p4, p5, p2sp1, p2sp2, p2sp3, p4sp1, p4sp2;
    p1 = new JPanel(); p2 = new JPanel(); p2sp1 = new JPanel();
    p2sp2 = new JPanel(); p2sp3 = new JPanel(); p3 = new JPanel();
    p4 = new JPanel(); p4sp1 = new JPanel(); p4sp2 = new JPanel(); p5 = new JPanel();

    space1 = (int) ( h*0.1);
    space2 = (int) (6.5*space1);
    space3 = 3*dis1 + 3*lfsz;

    p1.setBounds(0, 0, w, space1);
    p2.setBounds(0, space1, w, (int)(4.5*space1));
    p3.setBounds(0, (int)(5.5*space1), w, space1);
    p4.setBounds(0, space2, w, 2*space1);
    p5.setBounds(0, 9*space1, w, space1);

    p2sp1.setBounds(0, 0, w/3, 5*space1);
    p2sp2.setBounds(w/3, 0, w/3, 5*space1);
    p2sp3.setBounds(2*w/3, 0, w/3, 5*space1);
    p4sp1.setBounds(0, 0, w/2, 2*space1);
    p4sp2.setBounds(w/2, 0, w/2, 2*space1);
    p1.setLayout( new BorderLayout() );
    p3.setLayout( new BorderLayout() );
    p2.setLayout(null);
    p4.setLayout(null);

    Box p2b1 = Box.createVerticalBox();

```

```

Box p2b2 = Box.createVerticalBox();
Box p2b3 = Box.createVerticalBox();
Box p4b1 = Box.createVerticalBox();
Box p4b2 = Box.createVerticalBox();
Box p5b = Box.createHorizontalBox();
JLabel title1 = new JLabel("ISP Settings", JLabel.CENTER);
title1.setFont(f);
title1.setForeground(Color.darkGray);

p1.add(title1, "Center");
add(p1);

p2b1.add(Box.createVerticalStrut(dis1));
JLabel p2lb1 = new JLabel("Name of ISP", JLabel.CENTER);
p2lb1.setFont(f); p2b1.add(p2lb1);

JLabel p2lb2 = new JLabel("IP gateway", JLabel.CENTER);
p2lb2.setFont(f);
p2b1.add(Box.createVerticalStrut(dis1)); p2b1.add(p2lb2);

JLabel p2lb3 = new JLabel("IP address/Mask", JLabel.CENTER);
p2lb3.setFont(f);
p2b1.add(Box.createVerticalStrut(dis1)); p2b1.add(p2lb3);

JLabel p2lb4 = new JLabel("VP / VC value", JLabel.CENTER);
p2lb4.setFont(f);
p2b1.add(Box.createVerticalStrut(dis1)); p2b1.add(p2lb4);

JLabel p2lb5 = new JLabel("Physical ports", JLabel.CENTER);
p2lb5.setFont(f);
p2b1.add(Box.createVerticalStrut(dis1)); p2b1.add(p2lb5);
p2sp1.add(p2b1); p2.add(p2sp1);

nameField = new JTextField(col);
gateField = new JTextField(col);
ipField = new JTextField(col);
vpField = new JTextField(col);
nameField.setFont(f2); gateField.setFont(f2); ipField.setFont(f2);
vpField.setFont(f2);

p2b2.add(Box.createVerticalStrut(dis1)); p2b2.add(nameField);
p2b2.add(Box.createVerticalStrut(dis1)); p2b2.add(gateField);
p2b2.add(Box.createVerticalStrut(dis1)); p2b2.add(ipField);
p2b2.add(Box.createVerticalStrut(dis1)); p2b2.add(vpField);

String[] portStrings = { "Slot 0  J1", "Slot 1  J2", "Slot 2  J3", "Slot 3  J4" };
portList = new JComboBox(portStrings);
portList.setBounds(350, 100, 137, 35);
portList.addActionListener(this);
p2b2.add(Box.createVerticalStrut(dis1)); p2b2.add(portList);
p2sp2.add(p2b2); p2.add(p2sp2);

maskField = new JTextField(col);
vcField = new JTextField(col);
maskField.setFont(f2); vcField.setFont(f2);
p2b3.add(Box.createVerticalStrut(space3)); p2b3.add(maskField);
p2b3.add(Box.createVerticalStrut(dis1)); p2b3.add(vcField);
p2sp3.add(p2b3); p2.add(p2sp3);
add(p2);

```

```

JLabel title2 = new JLabel("RADIUS Authentication Server", JLabel.CENTER);
title2.setFont(f);
title2.setForeground(Color.darkGray);
p3.add(title2, "Center");
add(p3);

JLabel p4lb1 = new JLabel("IP address", JLabel.CENTER);
p4lb1.setFont(f);
p4b1.add(Box.createVerticalStrut(dis1)); p4b1.add(p4lb1);

JLabel p4lb2 = new JLabel("Shared secret", JLabel.CENTER);
p4lb2.setFont(f);
p4b1.add(Box.createVerticalStrut(dis1)); p4b1.add(p4lb2);

p4sp1.add(p4b1); p4.add(p4sp1);

radIPAddrField = new JTextField(col);
edgeField = new JTextField(col);
radIPAddrField.setFont(f2); edgeField.setFont(f2);
edgeField.setText(secret1);

p4b2.add(Box.createVerticalStrut(dis1)); p4b2.add(radIPAddrField);
p4b2.add(Box.createVerticalStrut(dis1)); p4b2.add(edgeField);
p4sp2.add(p4b2); p4.add(p4sp2);

add(p4);

cancel = new JButton("Cancel");
cancel.setBackground(new Color(192, 192, 255));
cancel.addActionListener(this);
cancel.setFont(new Font ("Dialog", Font.BOLD, bsize));
p5b.add(Box.createHorizontalStrut(2*dis2)); cancel.setBorder(border2);
p5.add(cancel);

JLabel empty = new JLabel(" ");
empty.setSize(w/10, bsize);

ok = new JButton("OK");
ok.setBackground(new Color(192, 192, 255));
ok.addActionListener(this);
ok.setFont(new Font ("Dialog", Font.BOLD, bsize));
p5b.add(Box.createHorizontalStrut(dis2)); ok.setBorder(border2);
p5.add(empty); p5.add(ok); add(p5);

} // end of C_IspPanel constructor

public void actionPerformed(ActionEvent e) {
    Object source = e.getSource();

    if(source == cancel)
        event = E_CANCEL;
    else if (source == ok) {

        if ( check() == false)
            return;

        ispObj = new IspObj();
        fill_IspObj();
        setCommands();
        event = E_OK;
    }
}

```

```

    }

} // end of ActionPerformed

} // end C_IspPanel

public void fill_IspObj() {
    ispObj.name = name;
    ispObj.ispIPAddr = ispIPAddr;
    ispObj.rtIPAddr = rtIPAddr;
    ispObj.rtMask = rtMask;
    ispObj.vp = vpI;
    ispObj.vc = vcI;
    ispObj.port = vpStr;
    ispObj.radIPAddr = radIPAddr;
    ispObj.edge = secret1;
}

// *****
//Validation check for finding empty String//

public boolean checkEmptyStr(String str) {
    if (str.length() == 0) {
        Toolkit.getDefaultToolkit().beep();
        p.popUpErrorMsg("Empty fields are invalid data. Please fill in the empty field(s)");
        return true;
    }
    return false;
}

public boolean check() {

    String ip1, ip3;
    String name, ispIPAddr, rtIPAddr, rtMask, vpStr, vcStr, radIPAddr, str8;
    ERROR_FOUND = false;
    name = nameField.getText().trim();
    ispIPAddr = gateField.getText().trim();
    rtIPAddr = ipField.getText().trim();
    rtMask = maskField.getText().trim();
    vpStr = vpField.getText().trim();
    vcStr = vcField.getText().trim();
    radIPAddr = radIPAddrField.getText().trim();
    str8 = edgeField.getText().trim();

    if ( checkEmptyStr(name) || checkEmptyStr(ispIPAddr) ||
        checkEmptyStr(rtIPAddr) || checkEmptyStr(rtMask) ||
        checkEmptyStr(vpStr) || checkEmptyStr(vcStr) ||
        checkEmptyStr(radIPAddr) || checkEmptyStr(str8)
    )
        return false;

    try { ip1 = InetAddress.getByName(ispIPAddr).getHostAddress(); }
    catch(UnknownHostException u) {
        Toolkit.getDefaultToolkit().beep();
        p.popUpErrorMsg(""+ispIPAddr+" is an invalid IP address. Please try again.");
        ERROR_FOUND = true;
    }

    try { ip1 = InetAddress.getByName(rtIPAddr).getHostAddress(); }
    catch(UnknownHostException u) {

```

```

Toolkit.getDefaultToolkit().beep();
p.popUpErrorMsg(""+rtIPAddr+" is an invalid IP address. Please try again.");
ERROR_FOUND = true;
}
////////////////////////////////////
String token;
StringTokenizer st = new StringTokenizer (maskField.getText(), ".");
for (int i = 0; i < 4; i++) {
    try {
        token = st.nextToken();
        int x = Integer.parseInt(token);
        if ( x < 0 || x > 255) {
            Toolkit.getDefaultToolkit().beep();
            p.popUpErrorMsg(""+rtMask+" is an invalid subnet mask. Please try again.");
            ERROR_FOUND = true;
            break;
        }
    }
    catch(Exception e) {
        Toolkit.getDefaultToolkit().beep();
        p.popUpErrorMsg(""+rtMask+" is an invalid subnet mask. Please try again.");
        ERROR_FOUND = true;
        break;
    }
}
////////////////////////////////////
try { vpI = Integer.parseInt(vpStr); }
catch(NumberFormatException e) {
    Toolkit.getDefaultToolkit().beep();
    p.popUpErrorMsg("Invalid integer. Please try again.");
    ERROR_FOUND = true;
}

if (vpI < MIN_NUM_VP2 || vpI > MAX_NUM_VP2) {
    Toolkit.getDefaultToolkit().beep();
    p.popUpErrorMsg("Invalid vp value. Default value is 0. Please try again.");
    ERROR_FOUND = true;
}

try { vcI = Integer.parseInt(vcStr); }
catch(NumberFormatException e) {
    Toolkit.getDefaultToolkit().beep();
    p.popUpErrorMsg("Invalid integer. Please try again.");
    ERROR_FOUND = true;
}

if (vcI < MIN_NUM_VC2) {
    Toolkit.getDefaultToolkit().beep();
    p.popUpErrorMsg("Invalid vc value. Minimum value is 32. Please try again.");
    ERROR_FOUND = true;
}

check_pvc();

try { ip1 = InetAddress.getByName(radIPAddr).getHostAddress(); }
catch(UnknownHostException u) {
    Toolkit.getDefaultToolkit().beep();
    p.popUpErrorMsg(""+radIPAddr+" is an invalid IP address. Please try again.");
    ERROR_FOUND = true;
}

```



```

if ( str8.equals(secret1) == false ) {
    Toolkit.getDefaultToolkit().beep();
    p.popUpErrorMsg(""+str8+" is not the correct secret. Please try again.");
    ERROR_FOUND = true;
}

if (ERROR_FOUND)
    return false;

this.name=name;
this.ispIPAddr=ispIPAddr;
this.rtIPAddr=rtIPAddr;
this.rtMask=rtMask;
this.vpStr=vpStr;
this.vcStr=vcStr;
this.radIPAddr=radIPAddr;
this.str8=str8;

return true;

} // end of check()

public void check_pvc() {

    boolean vpExists = false, newVpReq;
    int endVcVal, restVcVal, pvcsLeft, vpPos, count, pp;
    pp = portList.getSelectedIndex();
    Vp vp = new Vp(); Vc vc = new Vc();
    pvc[pp].trimToSize();
    for (int i = 0; i < pvc[pp].capacity(); i++) {
        vp = (Vp) pvc[pp].elementAt(i);
        if(vp.value == vpI) {
            vp.vcs.trimToSize();
            for (int j = 0; j < vp.vcs.capacity(); j++) {
                vc = (Vc) vp.vcs.elementAt(j);
                if (vc.value == vcI) {
                    Toolkit.getDefaultToolkit().beep();
                    p.popUpErrorMsg("PVC "+ vpI + "/" + vcI + " is occupied. Please try again.");
                    ERROR_FOUND = true;
                }
            }
            vpExists = true; vpPos = i;
            break;
        }
    }

    if (ERROR_FOUND)
        return;

    if (vpExists) {
        vc = new Vc(); vc.value = vcI;
        vp.vcs.addElement(vc);
        return;
    }

    vp = new Vp(); vc = new Vc();
    vp.value = vpI; vc.value = vcI;
    vp.vcs.add(vc); vp.vcs.trimToSize();
    pvc[pp].add(vp); pvc[pp].trimToSize();
}

```

```

}

public void setCommands() {

    int cmdIndex = 0;
    int selectNr = 1645;
    //////////////////////////////////////
    String selectPort = new String();

    switch (portList.getSelectedIndex()) {
        case 0 : selectPort = "J1" ;
                break;
        case 1 : selectPort = "J2" ;
                break;
        case 2 : selectPort = "J3" ;
                break;
        case 3 : selectPort = "J4" ;
                break;
    }

    String ip = new String();
    StringTokenizer st = new StringTokenizer (radIPAddrField.getText(), ".");
    for (int i = 0; i < 3; i++)
        ip = ip + st.nextToken() + "." ;
    ip = ip + "0";
    String initCmd1 = "ADD ACCESS PARTITION ENTRY ";
    String initCmd2 = "SET ACCESS PARTITION IP GATEWAY ";
    String initCmd4 = "ADD ATM CIP_PVC PVC V7.";
    String initCmd5 = "ADD SERVICE PROFILE ENTRY", sp = "SP_";
    String initCmd6 = "SET SERVICE PROFILE ACCESS PARTITION ";
    String initCmd7 = "SET SERVICE PROFILE MODEM POOL ", def = "default";
    String initCmd11 = "SET SERVICE PROFILE PORT PROFILE V7.";
    String initCmd13 = "ADD ATM INTERFACE VIRTUAL CHANNEL PVC", llcStr = "llc";
    String initCmd14 = "ADD IP NETWORK ENTRY ";
    String initCmd15 = "SET IP NETWORK MTU ";
    String initCmd16 = "ADD IP ROUTE ENTRY";
    String initCmd17 = "ADD RADIUS AUTHENTICATION SERVER ENTRY ";
    String initCmd18 = "ADD RADIUS ACCOUNTING SERVER ENTRY ";
    String initCmd = "CONTEXT";

    ispObj.cmd.add(initCmd1 + "\""+name+"\" ");
    ispObj.cmd.add(initCmd2 + "\""+name+"\" " + " " + ispIPAddr);
    ispObj.cmd.add(initCmd5 + " " + "\""+sp+name+"\"");
    ispObj.cmd.add(initCmd6 + " " + "\""+sp+name+"\" " + " " + "\""+name+"\"");
    ispObj.cmd.add(initCmd7 + " " + "\""+sp+name+"\" " + " " + "\""+def+"\"");
    ispObj.cmd.add(initCmd11 + ispNum + " " + "\""+sp+name+"\"");
    ispObj.cmd.add(initCmd13 + " " + selectPort + " " + vpI + " " + vcI +
        " " + "1" + " " + "1" + " " + llcStr + " " + "9188");
    ispObj.cmd.add(initCmd4 + ispNum + " " + vpI + " " + vcI);
    ispObj.cmd.add(initCmd + " " + name);
    ispObj.cmd.add(initCmd14 + rtIPAddr + " " + rtMask + " " + "V7." + ispNum);
    ispObj.cmd.add(initCmd15 + rtIPAddr + " " + "9180");
    ispObj.cmd.add(initCmd16 + " " + ip + " " + rtMask + " " + ispIPAddr + " " + "1");
    ispObj.cmd.add(initCmd17 + "1" + " " + radIPAddr + " " + "\""+secret1+"\" +
        " " + "5" + " " + "3" + " " + selectNr++);
    ispObj.cmd.add(initCmd18 + "1" + " " + radIPAddr + " " + "\""+secret1+"\" +
        " " + "5" + " " + "3" + " " + selectNr);

    ispObj.cmd.trimToSize();
} // end od setCommands

```

```
} // end of C_Isp
```

## **The Isp class**

```
package project;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.UIManager;
import javax.swing.border.*;
import java.lang.Thread;
import java.util.*;
import java.net.*;

public class Isp {

    public static final int S_INIT = 0;
    public static int state = S_INIT;

    public IspPanel ip;
    JButton cisp, eisp, disp;
    JComboBox cb1, cb2;
    Vector comboStrs = new Vector();
    public static final int E_NO_EVENT = 0;
    public static int event = E_NO_EVENT;
    public static int E_CISP = 1;
    public static int E_EISP = 2;
    public static int E_DISP = 3;
    public static int E_CANCEL = 4;

    public int waitForEvent() {
        int curEvent;
        while (event == E_NO_EVENT)
            try { Thread.sleep(100); } catch (InterruptedException e) {}
        curEvent = event;
        event = E_NO_EVENT;
        return curEvent;
    }

    public Isp(Rectangle R) {
        ip = new IspPanel(R);

        // edit and delete Isp's not implemented, hence diactivate buttons
        eisp.setEnabled(false);
        disp.setEnabled(false);
    }

    class IspPanel extends JPanel implements ActionListener {

        JLabel setting;
        JButton cancel;
        JComboBox jComboBox1, jComboBox2;
        private IspConfig ispFrame ;
        boolean ERROR_FOUND = false;
        String str = "ISP1";
        String selectPort;

        public IspPanel(Rectangle R) {
```

```

int w, h, lh, remh, lfsz, bfsz, dis1, dis2, dis3, bsize;
w = R.width;
h = R.height;

dis1 = h/10;
dis2 = 2 * dis1;
dis3 = 4 * dis1;

w = R.width;
h = R.height;
lh = (int) (h * 0.1);
remh = h - 2*lh;
lfsz = (int) h/24;
bfsz = (int) h/34;
setBounds(R);
setLayout(null);
Border border1, border2;
border1 = BorderFactory.createCompoundBorder(BorderFactory.createEtchedBorder
      (Color.white,new java.awt.Color(134, 134, 134)),
      BorderFactory.createEmptyBorder(5,5,5,5));

border2 = BorderFactory.createCompoundBorder(BorderFactory.createBevelBorder
      (BevelBorder.RAISED,Color.white,Color.white,new java.awt.Color(134, 134, 134),
      new java.awt.Color(93, 93, 93)),BorderFactory.createEmptyBorder(5,5,5,5));

JPanel p1 = new JPanel();
JPanel p2 = new JPanel();
JPanel p3 = new JPanel();
JPanel p4 = new JPanel();

p1.setLayout( new BorderLayout() );
Box b2 = Box.createVerticalBox();
Box b3 = Box.createVerticalBox();
p1.setBounds(0, 0, w, lh);

p2.setBounds(0, lh, w/2, remh);
p3.setBounds(w/2, lh, w/2, remh);
p4.setBounds(0, remh+lh, w, lh);

JLabel lb = new JLabel("ISP Configuration", JLabel.CENTER);
lb.setFont( new Font("Dialog", Font.BOLD, lfsz) );
lb.setForeground(Color.darkGray);

cisp = new JButton("Create ISP");
cisp.setBackground(new Color(192, 192, 255));
cisp.addActionListener(this);
cisp.setFont(new Font ("Dialog", Font.BOLD, bfsz));
b2.add(Box.createVerticalStrut(dis2)); cisp.setBorder(border2);
b2.add(cisp);

eisp = new JButton("Edit ISP");
eisp.setBackground(new Color(192, 192, 255));
eisp.addActionListener(this);
eisp.setFont(new Font ("Dialog", Font.BOLD, bfsz));
b2.add(Box.createVerticalStrut(dis1)); eisp.setBorder(border2);
b2.add(eisp);

disp = new JButton("Delete ISP");
disp.setBackground(new Color(192, 192, 255));
disp.addActionListener(this);

```

```

disp.setFont(new Font ("Dialog", Font.BOLD, bsize));
b2.add(Box.createVerticalStrut(dis1)); disp.setBorder(border2);
b2.add(disp);

cb1 = new JComboBox();
cb1.setBounds(350, 100, 137, 35);
cb1.addActionListener(this);
b3.add(Box.createVerticalStrut(dis2+2*dis1)); b3.add(cb1);

cb2 = new JComboBox();
cb2.setBounds(350, 100, 137, 35);
cb2.addActionListener(this);
b3.add(Box.createVerticalStrut(dis1)); b3.add(cb2);

cancel = new JButton("Cancel");
cancel.setBackground(new Color(192, 192, 255));
cancel.addActionListener(this);
cancel.setFont(new Font ("Dialog", Font.BOLD, bsize));
cancel.setBorder(border2); p4.add(cancel, "Center");
p1.add(lb, "Center");
p2.add(b2);
p3.add(b3);
add(p1); add(p2); add(p3); add(p4);
}

public void actionPerformed(ActionEvent e) {
    Object source = e.getSource();
    if(source == cisp)
        event = E_CISP;
    else if(source == eisp)
        event = E_EISP;
    else if(source == disp)
        event = E_DISP;
    else if(source == cancel)
        event = E_CANCEL;
}
}
}

```

## The PPPoATM class

```
package project;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.UIManager;
import javax.swing.border.*;
import java.lang.Thread;
import java.util.*;
import java.net.*;

public class PPPoATM {
    private int MAX_NUM_USERS = 2000;
    private int MAX_NUM_VP = 255;
    private int MAX_NUM_VC = 65535;
    private int MIN_NUM_VP = 0;
    private int MIN_NUM_VC = 32;
    boolean cancelPR = false;
    public static final int E_NO_EVENT = 0;
    public static final int E_CANCEL = 1;
    public static final int E_OK = 2;
    public int event = E_NO_EVENT;
    public int counter = 0;
    public String cmd[] = new String[2010];
    public Vector vcmd = new Vector();

    Vector[] pvc;

    public String str = new String();
    public boolean occPvcFound = false;

    public PPPoATMPanel pppoatmp;
    String user, vpStr, vcStr, portStr;

    PopUpError p = new PopUpError();
    Vector comboStrs = new Vector();
    JButton cancel, ok;
    JTextField userField, vpField, vcField;
    JComboBox portList, llcList;
    public int numOfUsers, vpI, vcI;

    public PPPoATM(Rectangle R, Vector[] pvc) { //Constructor
        pppoatmp = new PPPoATMPanel(R);
        this.pvc = pvc;
    }

    public int waitForEvent() {
        int curEvent;
        str = new String();
        occPvcFound = false;
        while (event == E_NO_EVENT)
            try { Thread.sleep(100); } catch (InterruptedException e) {}

        curEvent = event;
        event = E_NO_EVENT;
    }
}
```

```

return curEvent;
}

```

```

public void reset() {
    event = E_NO_EVENT;
    str = new String();
    occPvcFound = false;
}

```

```

class PPPoATMPanel extends JPanel implements ActionListener {

```

```

    JLabel setting;
    JButton ok, cancel;
    private IspConfig ispFrame ;
    boolean ERROR_FOUND = false;
    String selectPort;

```

```

public PPPoATMPanel(Rectangle R) {          ///Constructor of panel//
    Font f, f2;
    int w, h, lh, remh, lfsz, bfsz, dis1, dis2, dis3, bsize, col;
    int space1, space2;
    w = R.width;
    h = R.height;

    dis1 = h/40;
    dis2 = w/20;
    dis3 = 4 * dis1;

    w = R.width;
    h = R.height;
    lfsz = (int) (h/33);
    f = new Font("Dialog", Font.BOLD, lfsz);
    f2 = new Font("Dialog", Font.PLAIN, lfsz);
    bfsz = (int) (h/34);
    col = (int) (w/88);
    setBounds(R);
    setLayout(null);
    Border border1, border2;
    border1 = BorderFactory.createCompoundBorder(BorderFactory.createEtchedBorder
        (Color.white,new java.awt.Color(134, 134, 134)),
        BorderFactory.createEmptyBorder(5,5,5,5));

    border2 = BorderFactory.createCompoundBorder(BorderFactory.createBevelBorder
        (BevelBorder.RAISED,Color.white,Color.white,new java.awt.Color(134, 134, 134),
        new java.awt.Color(93, 93, 93)),BorderFactory.createEmptyBorder(5,5,5,5));

    JPanel p1, p2, p3, p2sp1, p2sp2, p2sp3;
    p1 = new JPanel(); p2 = new JPanel(); p2sp1 = new JPanel();
    p2sp2 = new JPanel(); p2sp3 = new JPanel(); p3 = new JPanel();

    space1 = (int) ( h*0.1);
    space2 = 3*dis1 + 3*lfsz;

    p1.setBounds(0, 0, w, space1);
    p2.setBounds(0, 3*space1, w, 4*space1);
    p3.setBounds(0, 9*space1, w, space1);

    p2sp1.setBounds(0, 0, w/3, 4*space1);
    p2sp2.setBounds(w/3, 0, w/3, 4*space1);
    p2sp3.setBounds(2*w/3, 0, w/3, 4*space1);

```

```

p1.setLayout( new BorderLayout() );
p2.setLayout(null);

Box p2b1 = Box.createVerticalBox();
Box p2b2 = Box.createVerticalBox();
Box p2b3 = Box.createVerticalBox();

JLabel title1 = new JLabel("PPP over ATM Configuration", JLabel.CENTER);
title1.setFont(f);
title1.setForeground(Color.darkGray);
p1.add(title1, "Center");
add(p1);

p2b1.add(Box.createVerticalStrut(dis1));
JLabel p2lb1 = new JLabel("ATM Card Position", JLabel.CENTER);
p2lb1.setFont(f);
p2b1.add(p2lb1);

JLabel p2lb2 = new JLabel("Number of Users", JLabel.CENTER);
p2lb2.setFont(f);
p2b1.add(Box.createVerticalStrut(dis1)); p2b1.add(p2lb2);

JLabel p2lb3 = new JLabel("Start VP / VC value", JLabel.CENTER);
p2lb3.setFont(f);
p2b1.add(Box.createVerticalStrut(dis1)); p2b1.add(p2lb3);

JLabel p2lb4 = new JLabel("Choose llc or vc_mux", JLabel.CENTER);
p2lb4.setFont(f);
p2b1.add(Box.createVerticalStrut(dis1)); p2b1.add(p2lb4);
p2sp1.add(p2b1); p2.add(p2sp1);

userField = new JTextField(col);
vpField = new JTextField(col);
vcField = new JTextField(col);

String[] portStrings = { "Slot 0   J1", "Slot 1   J2", "Slot 2   J3", "Slot 3   J4" };
portList = new JComboBox(portStrings);
portList.setBounds(350, 100, 137, 35);
portList.addActionListener(this);
p2b2.add(Box.createVerticalStrut(dis1)); p2b2.add(portList);

userField.setFont(f2); vpField.setFont(f2); vcField.setFont(f2);
p2b2.add(Box.createVerticalStrut(dis1)); p2b2.add(userField);
p2b2.add(Box.createVerticalStrut(dis1)); p2b2.add(vpField);

String[] llcStrings = { "llc", "vc_mux" };
llcList = new JComboBox(llcStrings);
llcList.setBounds(350, 300, 137, 35);
llcList.addActionListener(this);
p2b2.add(Box.createVerticalStrut(dis1)); p2b2.add(llcList);
p2sp2.add(p2b2); p2.add(p2sp2);

p2b3.add(Box.createVerticalStrut(space2)); p2b3.add(vcField);
p2sp3.add(p2b3); p2.add(p2sp3);
add(p2);

cancel = new JButton("Cancel");
cancel.setBackground(new Color(192, 192, 255));
cancel.addActionListener(this);

```



```

cancel.setFont(new Font ("Dialog", Font.BOLD, bfsize));
cancel.setBorder(border2); p3.add(cancel);

JLabel empty = new JLabel("      ");
empty.setSize(w/10, bfsize);
p3.add(empty);

ok = new JButton("OK");
ok.setBackground(new Color(192, 192, 255));
ok.addActionListener(this);
ok.setFont(new Font ("Dialog", Font.BOLD, bfsize));
ok.setBorder(border2); p3.add(ok);
add(p3);

} // end of PPPoATMPanel constructor

public void actionPerformed(ActionEvent e) {
    Object source = e.getSource();

    if (source == cancel)
        event = E_CANCEL;

    else if (source == ok) {
        if ( check() == false )
            return;
        if (occPvcFound == false)
            setCommands();
        event = E_OK;
    }

} //end of actionPerformed
} // end PPPoATMPanel

public boolean update_pvcs() {
    boolean vpExists = false, newVpReq;
    int endVcVal, restVcVal, pvcsLeft, vpPos, count = 0, pp, p, c;
    p = vpI; c = vcI;
    pvcsLeft = numOfUsers;
    pp = portList.getSelectedIndex();
    Vp vp = null; Vc vc = new Vc();

    // find vp
    pvc[pp].trimToSize();
    for (int i = 0; i < pvc[pp].capacity(); i++) {
        vp = (Vp) pvc[pp].elementAt(i);
        if(vp.value == p) {
            vpExists = true;
            break;
        }
    }

    // if not found, create a new one for vpI value
    if (vpExists == false) {
        vp = new Vp();
        vp.value = p;
        pvc[pp].add(vp);
    }

    int del = 1;
    while(pvcsLeft > 0) {

```

```

pvc[pp].trimToSize(); vp.vcs.trimToSize();

if (c > MAX_NUM_VC) {
    p++; c = MIN_NUM_VC;
    vpExists = false;

    for (int i = 0; i < pvc[pp].capacity(); i++) {
        vp = (Vp) pvc[pp].elementAt(i);
        if(vp.value == p) {
            vpExists = true;
            break;
        }
    }

    // if not found, create a new one for vpI value
    if (vpExists == false && occPvcFound == false) {
        vp = new Vp();
        vp.value = p;
        pvc[pp].add(vp);
        pvc[pp].trimToSize();
    }
}

// occupy pvc
if (vpExists == false && occPvcFound == false) {
    vc = new Vc(); vc.value = c;
    vp.vcs.add(vc); vp.vcs.trimToSize();
    pvcsLeft--; c++;
    continue;
}

// go through vc's for this vp
for (int j = 0; j < vp.vcs.capacity(); j++) {
    vc = (Vc) vp.vcs.elementAt(j);
    // occupied pvc found
    if (vc.value == c) {
        str = str + p + "/" + c;
        if (count > 20) {
            str = str + "\n";
            count = 1;
        }
        else {
            str = str + ", ";
            count++;
        }
        occPvcFound = true;
        break;
    } // end of occupied pvc found

} // next Vc

if (occPvcFound == false) {
    vc = new Vc(); vc.value = c;
    vp.vcs.add(vc); vp.vcs.trimToSize();
}
pvcsLeft--; c++;
} // end of while

```

```

        if (occPvcFound == true)
            return false;
        return true;
    }

    public boolean check() {

        String user = userField.getText().trim();
        if (user.length() == 0) {
            p.popUpErrorMsg("! Empty fields are invalid data. Please select the number of
users");

            return false;
        }
        try {
            numOfUsers = Integer.parseInt( user );
        }
        catch(NumberFormatException e) {
            p.popUpErrorMsg("Invalid integer. Please try again.");
            return false;
        }
        if (numOfUsers > MAX_NUM_USERS) {
            p.popUpErrorMsg("Number of users should be up to 2000. Please try again.");
            return false;
        }
        ///////////////////////////////////////////////////////////////////
        String vpStr = vpField.getText().trim();
        if (vpStr.length() == 0) {
            p.popUpErrorMsg("! Empty fields are invalid data. Please try again");
            return false;
        }
        try {
            vpI = Integer.parseInt( vpStr );
        }
        catch(NumberFormatException e) {
            p.popUpErrorMsg("Invalid integer. Please try again.");
            return false;
        }
        if (vpI < MIN_NUM_VP || vpI > MAX_NUM_VP) {
            p.popUpErrorMsg("Invalid vp value. Default value is 0. Please try again.");
            return false;
        }

        String vcStr = vcField.getText().trim();
        if (vcStr.length() == 0) {
            p.popUpErrorMsg("! Empty fields are invalid data. Please try again");
            return false;
        }
        try {
            vcI = Integer.parseInt( vcStr );
        }
        catch(NumberFormatException e) {
            p.popUpErrorMsg("Invalid integer. Please try again.");
            return false;
        }
        if (vcI < MIN_NUM_VC) {
            p.popUpErrorMsg("Invalid vc value. Default value is 32. Please try again.");
            return false;
        }
        }

        int vcChk;
    }

```

```

        vcChk = MAX_NUM_VC - vcI;
        if (vpI == MAX_NUM_VP && vcChk < numOfUsers) {
            p.popUpErrorMsg("No more free pvcs. Please select a lower value for vp or vc.");
            return false;
        }

        update_pvcs();

        this.user=user;
        this.vpStr=vpStr;
        this.vcStr=vcStr;
        return true;
    } // end of check()

public void setCommands() {

    String initCmd1 = "ADD ATM INTERFACE VIRTUAL CHANNEL PVC ";
    String initCmd2 = "SET VIRTUAL PORT PHYSICAL PORT V4.";
    String initCmd3 = "ADD ATM PPP PVC V4.";
    String initCmd4 = "ADD SERVICE PROFILE ENTRY", cd = "CDNR";
    String initCmd5 = "SET SERVICE PROFILE ACCESS PARTITION",
        id = "UID- SELECT";
    String initCmd6 = "SET SERVICE PROFILE MODEM POOL", def2 = "default" ;
    String initCmd7 = "SET SERVICE PROFILE PORT PROFILE V4.";

    int pvcSelect = numOfUsers + vcI;
    int cmdIndex = 0;
    int k = 1;
    int m = 1;
    int virt2 = 1;
    String atm = "ATMSELECT";
    String command;
    String selectPort = new String();

    switch (portList.getSelectedIndex()) {
        case 0 : selectPort = "J1" ;
            break;
        case 1 : selectPort = "J2" ;
            break;
        case 2 : selectPort = "J3" ;
            break;
        case 3 : selectPort = "J4" ;
            break;
    }

    vcmd.add(initCmd4 + " " + "\"" + atm + "\"" + " " + cd);
    vcmd.add(initCmd5 + " " + "\"" + atm + "\"" + " " + "\"" + id + "\"");
    vcmd.add(initCmd6 + " " + "\"" + atm + "\"" + " " + "\"" + def2 + "\"");

    for ( int j = vcI ; j < pvcSelect; j++) {
        command = initCmd7 + virt2++ + " " + "\"" + atm + "\"";
        vcmd.add(command);
        vcmd.trimToSize();
    }

    for ( int j = vcI ; j < pvcSelect; j++) {
        command = initCmd1 + selectPort + " " + vpStr + " " + j + " " +
            "1" + " " + "1" + " " + llcList.getSelectedIndex() + " " + "9188";
        vcmd.add(command);
    }
}

```

```

    } // end of for loop

        for ( int j = vcI ; j < pvcSelect; j++) {
            command = initCmd2 + m+++ + " " + selectPort;
            vcmd.add(command);
        } // end of for loop

        for ( int j = vcI ; j < pvcSelect; j++) {
            command = initCmd3 + k+++ + " " + vpI + " " + j;
            vcmd.add(command);
        } // end of for loop
    vcmd.trimToSize();
} //end of setCommands
}

```

## **The IspObj class**

```

package project;

import java.util.Vector;

public class IspObj {
    public String name, ispIPAddr, rtIPAddr, rtMask, port;
    public String radIPAddr, edge;
    public int vp, vc;
    public Vector cmd = new Vector();
    public int counter = 0;
}

```

## **The Router class**

```

package project;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

import java.util.Vector;

public class Router {

    public static void main(String[] args) {

        Handler hr = new Handler();
        hr.run();
    }
}

```

### **The Vc class**

```
package project;

import java.util.Vector;

public class Vc {

    public int value;

}
```

### **The Vp class**

```
package project;

import java.util.Vector;

public class Vp {

    public int value;
    public Vector vcs;

    public Vp() {
        vcs = new Vector();
    }

}
```

### **The Txt class**

```
package project;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.util.*;
import javax.swing.border.*;

class Txt extends JPanel {
    public TextArea t;

    public Txt(Rectangle R) { //Constructor//
        int w = R.width, h = R.height;
        Border border1;
        setBounds(R);
        border1 = BorderFactory.createCompoundBorder(BorderFactory.createEtchedBorder
            (Color.white,new java.awt.Color(134, 134, 134)),
            BorderFactory.createEmptyBorder(10,10,10,10));
        setBorder(BorderFactory.createLineBorder(Color.black));

        t = new TextArea(h/21, (w/8) + w/100);
        add(t);
    }
}
```

```
}
```

## **The Welcome class**

```
package project;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class Welcome extends JPanel {

    public Welcome(Rectangle R) {

        Border border1;
        border1 = BorderFactory.createCompoundBorder(BorderFactory.createEtchedBorder
            (Color.black,new java.awt.Color(134, 134, 134)),
            BorderFactory.createEmptyBorder(10,10,10,10));
        setBorder(BorderFactory.createLineBorder(Color.black));

        setBounds(R);
        setLayout( new BorderLayout() );
        JLabel msg = new JLabel("Access Setup Wizard", JLabel.CENTER);
        msg.setFont( new Font("Dialog", Font.BOLD, R.height/15) );
        msg.setForeground(Color.blue);
        add(msg, "Center");
    }

}
```

## **The PopUpError class**

```
package project;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class PopUpError {
    JFrame fr = new JFrame();

    public void popUpErrorMsg(String errorMsg) {
        JOptionPane.showMessageDialog(fr, errorMsg,
            "Error message", JOptionPane.ERROR_MESSAGE);
    }

    public void popUpInformationMsg(String errorMsg) {
        JOptionPane.showMessageDialog(fr, errorMsg,
            "Information", JOptionPane.INFORMATION_MESSAGE);
    }

}
```