Computer Science

Ante Engström, Rassol Raissi

Firewall Configuration System

Bachelor's Project

2002:20

This report is submitted in partial fulfillment of the requirements for the Bachelor’s degree in Computer Science. All material in this re port which is not our own work has been identified and no material is included for which a degree has previously been conferred.

Ante Engström

Rassol Raissi

Approved, June 5, 2002

Advisor: Stefan Alfredsson

Examiner: Stefan Alfredsson

Abstract

This document describes our Bachelor’s Project. We have created an application to improve configuration of certain firewalls for Internet Security Systems (ISS). The project consists of an application and a database. The application is named Firewall Configuration System (FCS).

Firewalls in general are introduced, to give an understanding of the background of our work. The design and implementation for the GUI and the database are then described in detail, which will give the reader an insight of how the application functions and how the project is constructed.

The application is programmed using Microsoft Visual C++ and the database runs on a Microsoft SQL Server.

Contents

List of Figures

List of tables

1 Introduction

The Internet grows day by day and so does the threat of being mistreated when being connected. Hazards like viruses or persons trying to gain access to ones computer cannot be disregarded. To be connected to the Internet is essential for organizations of this modern age. Fortunately there are ways to protect private information stored on their computers. The most common way is to set up a firewall between the internal network and the Internet. This means that all outgoing and incoming traffic will pass through the firewall so it can be controlled before it is sent on. A firewall is a clever way of protecting their data but it needs maintenance. Instead of having to educate people to maintain the firewall, the organizations' turn to businesses like Internet Security Systems (ISS) who has qualified personnel to deal with this.

When configuring a firewall, ISS does not send a person over to the organization to do this because the firewall can be located abroad. To maintain the firewall they set up a virtual private network (VPN, secure communication) between the firewall and themselves. Through the VPN they can configure the firewall from their own office, by sending commands to the firewall.

Here is where this project starts. At this moment configurations are set manually, which is time consuming and mistakes can easily be made. The task given was to construct an application with a purpose to make the procedure of writing these configuration files easier. This application should support two small office firewalls, WatchGuard SOHO and NetScreen-5XP. Two parts build up the application; one client containing a graphical user interface (GUI) and a database running on a server. The database should be able to handle several connections simultaneously and the GUI should run under Windows NT and interact with the database through ODBC (see chapter 2.6).

The rest of the report is organized as follows:

Chapter 2. Gives an introduction to the project's initiator, Internet Security Systems, what firewalls are and how they work. Also a short overview of VPN, IPSec, Watchguard SOHO, NetScreen-5XP and ODBC is given here.

Chapter 3. This is where the project specification, discussions and assumptions are. How and why we approached problems in a certain way.

Chapter 4. In this chapter the design of the application is discussed. First is the designing of the database then the designing of the GUI described. Last are a short overview of the classes that are used and a class diagram.

Chapter 5. Here is where the implementation is placed along with the problems and solutions that occurred during the development.

Chapter 6. This is where the test and evaluation is given for this project.

Chapter 7. The summary and final conclusions are given here.

2 Background

This chapter explains basic facts that are useful to know about when studying this project.

2.1 VPN & IPSec

The purpose of *VPN*s is to send data from one point to another inside a secure and efficient tunnel (see Figure 2.1). It carefully guards both ends of the tunnel so that only authorized users and their data can enter. The VPN encrypts data that is sent and decrypts receiving data. Furthermore, a VPN has security features that can limit users' access to certain sections only.

VPN is put up between different networks so that it seems like they are all on one big private network. This way people located on one of the networks can reach data on the other in a secure way.

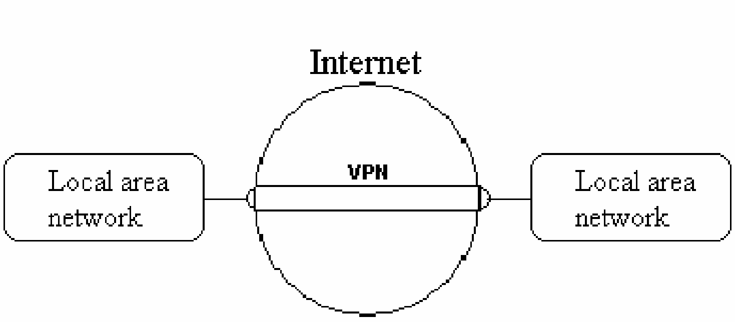


Figure 2.1: Vpn connection between two networks

Packets sent over the Internet should be protected if they contain private information. One way of doing this is by using *IPSec*. IPSec uses two mechanisms for ensuring the safety of packets. These two are the IPSec Authentication Header (AH) and the IPSec Encapsulating Security Payload (ESP). The first one digitally signs the outbound packet, both data payload and headers, with a hash value appended to the packet, verifying the identity of the source and destination machines and the integrity of the payload.

The second mechanism guarantees the integrity and confidentiality of the data in the original message by combining a secure hash and encryption of either the original payload by itself, or the headers and payload of the original packet.

2.2 Firewalls

With the increasing popularity of always -on connections, such as cable modems and DSL lines, most organisations and even home users are connected to the Internet 24 hours a day. This raises serious security concerns. Internet users need to be increasingly aware of security issues, as network traffic coming into the computer can cause damage to files and programs. There also exist intruders that want to break into computers or networks so that they can steal or alter any information that they can get their hands on. For example, the loss of financial records, e -mail, customer files and so on, can be devastating to an organisation or to an individual.

Installing a firewall is a good way of protecting the computer of a home user or the network of an organization.

2.2.1 Firewall Basics

A firewall is either only a software application (Personal firewall) or it is a combination between hardware and software (Hardware firewall) that is placed between the Internet connection and the computer or internal network. A firewall separates a computer or an internal network from the Internet, inspecting packets of data as they arrive at either side of the firewall (from the internal network or computer, or from the Internet) to determine whether it should be allowed to pass or be blocked. The firewall determines this by checking a list that contains all the rules that it must follow. The firewall administrator establishes these rules.

2.2.2 General techniques

Firewalls uses four general techniques to control access and enforce the site’s security policy.

Service control, this technique is used to see which types of Internet services can be accessed, inbound or outbound.

Direction control, this technique is used by the firewall to check which direction certain service requests may be initiated and allowed to flow through the firewall.

User control, the firewall uses this technique to check if a certain user has the right to access a certain service. This technique is mostly applied on users sitting inside the internal network but it may also be applied on users being outside the internal network. IPSec is used to authenticate the user that is sending the incoming traffic so it can pass the firewall.

Behaviour control, this technique is used for example, to reduce the information on a local web server seen by external access, or to set the firewall to filter e-mail to eliminate spam.

2.2.3 Why should an organisation employ firewalls?

An organisation should use firewalls in means of keeping unauthorised persons off the system. These intruders can cause a lot of damage to an organisation.

These persons may work for a competitor organisation that may want get information about future coming products, trade secrets, marketing strategies, or financial analysis.

They might be persons wanting to delete or change information just for the fun of it. By doing this they can change the appearance of an organisations web server, which may be seen by thousands of people in a matter of minutes. Things like this can damage the organisations reputation.

As we can see, it is in the organisations best interest to invest in a firewall protecting system.

2.2.4 Drawbacks of using firewalls

Firewalls are remarkable when it comes to protecting data behind the firewall, but there are some attacks that firewalls cannot handle, such as interception of mail and eavesdropping (intentional interception of data along the Internet).

As we know firewalls provide a single point of security and audit. This means that if an intruder gets through the firewall, he or she may have an opportunity to do anything they want to the system including stealing and altering information.

Another situation to consider is that an unsatisfied employee with vast knowledge of the organization can do the organization a great harm. Since the user is located on the inside of the firewall, there are no ways of preventing this employee to alter or to give away any information concerning the firm/organization.

2.2.5 Selecting firewalls

Home users usually choose the personal firewall because they only have one or a few computers that are connected to the Internet. This is a low cost alternative for protecting private information kept on their computers. There are some freeware/shareware personal firewalls that can be downloaded from the Internet, but these are often just a reduced version of the full ones. Of course one will have to pay for the full version personal firewalls. The full version firewalls might have better support possibilities and configuration management.

Organizations normally use hardware firewalls. They usually have a number of computers connected to the Internet making them able to do their work. Instead of having to install and administrate software firewalls on each and every one of these computers, they will probably choose to install the hardware firewall. The hardware firewall is placed between the organizations internal network and the Internet. By doing this, the administrator only has one firewall to configure and maintain.

As we can se f irewall applications vary in quality and cost. It is good then to consider the following points when selecting a firewall:

- Ease of installation/configuration.
- Does the firewall run without user intervention?
- Are there parameters that have to be set, and is it easy to do?
- Is there online help or technical support available?
- Does the firewall provide audit reports identifying time, location and type of attack?
- Is the cost of the firewall appropriate to the size of your business/office?
- Are maintenance/monitoring requirements suitable for the size and type of business?
- Will the firewall have a significant impact on the operation of the system as a whole?

There are a number of firewall products available with varying feature capabilities and prices. Home users who just have one computer connected to the Internet should consider the personal firewall, which is cheaper and easier to install. Offices with a number of computers connected to the Internet should choose to use hardware firewalls. These are slightly more expensive but offer wider protection for the organization.

2.3 ISS – Firewall Outsourcing

Founded in 1994, Internet Security Systems is a pioneer and world leader in software and services that protect corporate and personal information from an ever -changing spectrum of online threats and misuse. As organizations increasingly move operations online, the number and sophistication of threats to the networks, servers and desktops that empower these initiatives also continue to escalate. Internet Security Systems' solutions dynamically detect, prevent and respond to these threats.

Internet Security Systems' market includes any organization or individual with online digital assets to protect. Internet Security Systems is the trusted security provider for over 9,000 corporate customers, including 49 of the Fortune 50, the 10 largest U.S. securities firms, 10 of the world's largest telecommunications companies and major agencies and departments within U.S. local, state and federal governments.

Some examples of ISS Solutions and services are the RealSecure Protection System, BlackICE intrusion protection software, Managed Security Services (MSS) and ISS X-Force™.

The RealSecure Protection System software platform comprises integrated, centrally managed security assessment, intrusion detection and response, and enterprise decision-support functionality. The BlackICE intrusion protection comprises software solutions for small offices, and home offices deliver easily administered protection for any online assets. A managed Security Services (MSS) offering allows customers to focus on core business initiatives while leveraging ISS expertise to assess, design, deploy, manage and educate. The Internet Security Systems X -Force™ organization, an industry -leading security research and development organization, ensures that Internet Security Systems proactively stays on top of the latest security threats.

Internet Security Systems is headquartered in Atlanta, GA, with operations throughout the Americas, Asia, Australia, Europe and the Middle East.

The part of ISS, MSS EMEA, with offices in Brussels, Helsingborg and Karlstad that this project is involved in, provides the service MSS (Manage Security Services) for the EMEA market: The service includes management of the customer's security equipment. The customer owns the equipment in most cases and ISS applies support and maintenance to the infrastructure that is required to handle the customer's equipment. Depending on agreement they can also offer a "Customer portal" where the customer can fetch reports, read audits, order changes and so on.

One service provided by ISS is called "Small Office VPN". This service means that ISS sets up VPNs and monitors them for the client. Here it is VPN service in first hand which sales rather than firewall functionality. Furthermore some customers may wish to have a pure firewall service. To meet the customer requirements, ISS supports products that contain both firewall and VPN functionality. Two of these are Netscreen-5XP and Watchguard SOHO. These are used particularly in smaller offices (approximately 10 workstations) with a VPN connection to their head office. ISS handles the configuration by sending a configuration file over VPN to the firewall.

.

2.4 Watchguard SOHO

Figure 2.2: Watchguard SOHO

The Watchguard SOHO (Figure 2.2) is a security -dedicated hardware appliance that is easily installed between a ADSL or ISDN router and the network. It supports all leading operating systems. ISS uses the file transfer protocol (FTP) to configure this firewall by sending the configuration file directly to the device via a VPN.

Selected Key Benefits

- **Internet Security.** Protect all of your networked computers with dynamic stateful packet filtering firewall technology. Create filter rules based on port and protocol for both inbound and outbound traffic.
- **Easy Installation.** This plug -and-play security dedicated hardware device configures easily using any standard browser or file ftp.
- **Broadband Internet Sharing.** Share a single cable, DSL or ISDN high -speed Internet connection with up to 50 computers and save the cost of multiple connections.
- **Network Computers.** Network up to 50 computers to exchange e -mail and files, and to share a broadband Internet connection, printers and other equipment.
- **Branch Office VPN.** Establish a private, encrypted VPN tunnel with another location with the Firebox SOHO|tc. Branch office VPN is optional with the Firebox SOHO and may be added at any time.
- **Mobile User VPN Option.** Establish DES or 3DES-encrypted VPN tunnel with travelling users.

2.5 NetScreen-5XP

Figure 2.3: NetScreen

The NetScree n-5XP (Figure 2.3) is an Internet security appliance integrating firewall, virtual private networking (VPN) and traffic shaping functionality. With the VPN functionality built in, all management can be encrypted for truly secure remote management. It features wire -speed Ethernet performance for remote offices and telecommuters. The NetScreen-5XP is offered in two versions, one that allows 10 users and one that allows an unrestricted number of users.

ISS uses the command line interface (CLI) accessible in -band via SSH to configure this firewall.

Ssh (Secure Shell) is a program to log into another computer over a network, to execute commands in a remote machine, and to move files from one machine to another. It provides strong authentication and secure communications over unsecure channels.

- **Internet Security.** The NetScreen -5XP is fully capable of securing a broadband telecommuter or a small office. It has a fully integrated solution with security -optimized hardware, operating system and firewall, which provides higher level of security than patched-together software-based solutions.
- **Easy Installation and Managing.** Installing and managing appliances is easily accomplished using a built -in WebUI, command line interface, or NetScreen’s centr al management solutions.
- **VPN.** The NetScreen-5XP has a VPN solution supporting site -to-site and remote -access VPN applications. It has 3DES, DES and AES encryption using digital certificates, IKE auto-key, or manual key. SHA-1 and MD5 strong authentication
- **Traffic management**. Traffic management allows a network administrator to monitor, analyze, and allocate bandwidth utilized by various types of network traffic in real time, helping to ensure that web surfing or other non -critical applications do not impact business-critical traffic.

2.6 ODBC

Open DataBase Connectivity is a standard database access method developed by Microsoft. The goal of ODBC is to make it possible to access any data from any application, regardless of which database management system (DBMS) is handling the data. ODBC manages this by inserting a middle layer, called a database driver, between an application and the DBMS. The purpose of this layer is to translate the application's data queries into commands that the DBMS understands. To make this functional, both the application and the DBMS must be ODBC -compatible. Meaning the application must be capable of issuing ODBC commands and the DBMS must be capable to respond.

3 Project Specification

Some firewalls are configured by a text file consisting of several commands. This applies for example to Netscreen -5XP and Watchguard SOHO. Handling this manually is time consuming and allows human mistakes. It is also hard to get a good overview of changes to the configuration.

To improve this configuration procedure an application is appropriate. This application should handle parameter information for certain firewalls and then use the information to generate a configuration file consisting of commands in plain ASCII text format.

The application should be based upon a database so that different clients can access and configure simultaneously. Furthermore, the database should be constructed per customer to make it easy to get an overview over all the equipments and configurations for each customer. A copy of the actual configuration file need not be stored in the database.

Syntax may change on version updates so it should be easy to change the translation from parameter information to configuration file. The configurations for each firewall are mostly the same for each customer. The only thing that differs is firewall specific information like IP, VPN and passwords. Some kind of parameter data pattern can be used. But in some cases firewall configurations can break the pattern.

The assignment is limited to only concern Internet Security Systems Small Office service, see service definition appendix E ”Small Office, Managed Firewall Service”. The equipment that should be supported is Watchguard SOHO [5] and Netscreen-5XP [6].

The application must execute on Windows NT workstations and MS SQL Server should serve as the database server. MS Visual C++ was found to be appropriate as the tool for building the application.

4 Design

The work with designing the user interface felt as a good starting point to easier get an overview of how the application would look like and function. This was more difficult to accomplish without knowing how the database was to be structured, what data to be stored and in which tables. Designing the database structure was obviously this projects first challenge.

This chapter handles the database and the user interface along with the code construction.

4.1 The Database

The requirements concerning the database part of the project:

- Multiple clients are to be allowed access to the database at the same time.
- Microsoft SQL Server is to be used
- The database is to be structured per customer, which makes it easier to apply equipment and configuration for each customer.
- Users should be logged when entering the application and when adding or modifying a customer or firewall.
- The syntax used by the configuration file can be changed, so it should be easy to change parameter information.
- Every firewall has a pattern for its configuration file. ISS gives the same configuration to all customers. Every parameter has the same variables for each customer of a certain type except customer specific variables, for example IP-addresses. But there are exceptions when a firewall configuration does not follow the pattern, when one or more variables differ. We have to take this into consideration when programming and constructing the database. Parameters and variables are explained later on in this section.

4.1.1 Multiple clients, MS SQL Server & ODBC

Allowing multiple clients to access the database is no problem using ODBC and MS SQL Server. The application only needs to connect to an ODBC source, which handles the communication with the SQL Server database. An ODBC source must be defined for the application to function properly. This can be done in the Windows control panel. When an ODBC connection is defined it is directed to the server and given a name. This name is important, for it is the name that the application uses to find the right ODBC source.

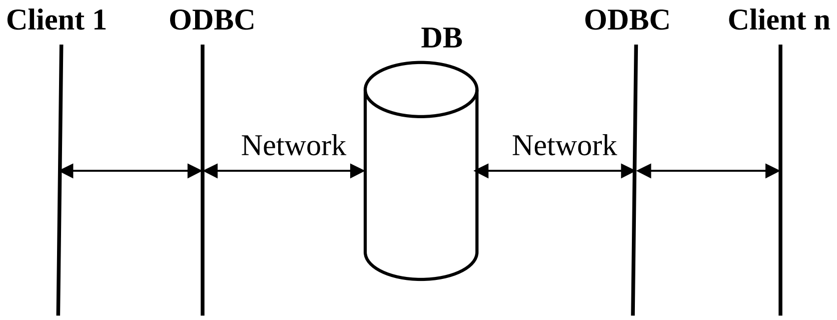


Figure 4.1: Multiple clients connecting to server via ODBC

4.1.2 The Structure. Tables & Relations

Since the database should be constructed with the customer as a base we started with creating the Customer table (Figure 4.2). Since every customer name is unique, CustomerName was chosen as the primary key.

CustomerName

Figure 4.2: Customer table

(Bold text shows the primary key). Every customer can use zero or more firewalls so a table called Firewall was created (Figure 4.3). In this table all specific information concerning the actual firewall is stored and what type of firewall it is. (WatchGuard SOHO, Netscreen-5XP…). Observe that one customer can be registered to two different firewalls. VPN information is also a necessary part of each firewall. This information consists of Remote

Network, Remote Net mask, Remote Gateway IP and Shared Secret. It is uncertain how many different VPN configurations that are needed for each firewall, so a new table is added for the VPN information.

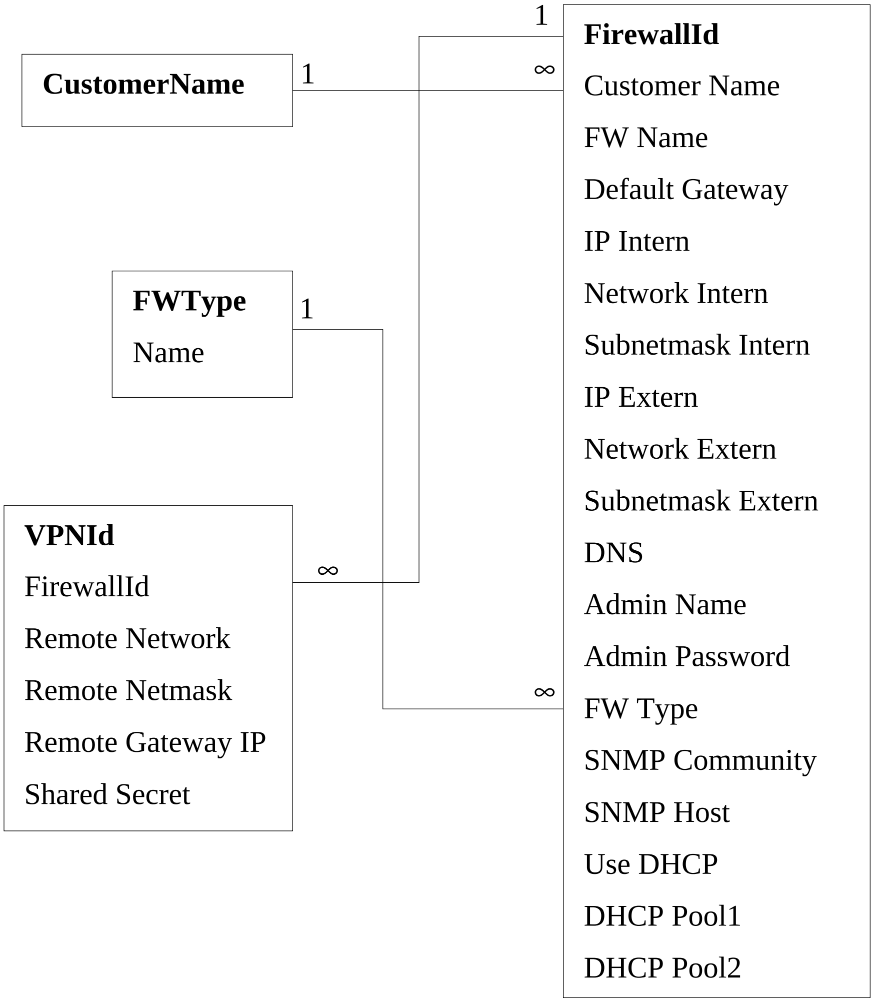


Figure 4.3: Tables Customer, Firewall, FirewallType & VPN

Since our program also need to be able to show who has generated configuration files and when this was done for each firewall, this also had to be structured in the database. To solve this all users must pass through authentication to get access to the application so they can be tracked. Two tables need to be created (Figure 4.4). A user table with user information, such as the username and password, and a log table containing the tracked users. Information that is needed in the log table is the user and firewall concerned, and the data along with the event that was logged. There are five events, which need to be logged:

- User logged in successfully.
- User adds a customer.
- User renames a customer.
- User adds a firewall to a customer.
- User edits a certain firewall.

The login routine is used to prevent users to access the application, not to prevent access to the database. Anyone can access the database if a proper ODBC connection is set up. The application also needs authentication to be able to log the user.

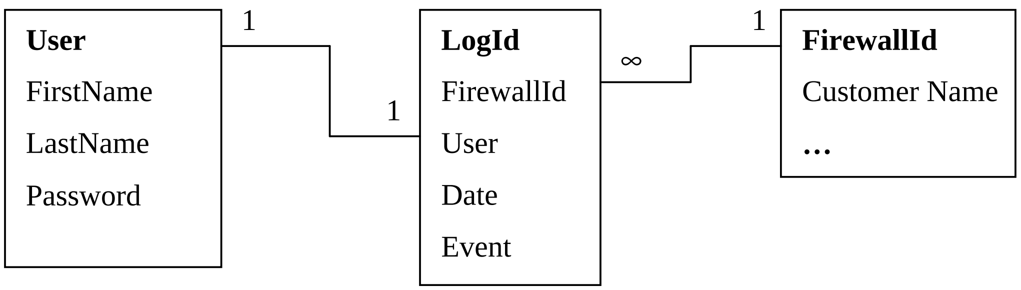


Figure 4.4: Tables User, Log & FireWall

The following parts of this section deal with how to store the information used to build up the configuration files. The database should be constructed to be flexible to manage configuration changes that might occur on version updates.

A command line in a configuration file consists of two parts. The line starts with a *parameter*, which is similar to a data type. A parameter follows by one or more *variables* that conclude the command line. A quick look at the configuration files (appendix A & B) for both firewalls shows this. This was easily designed (see Figure 4.5).

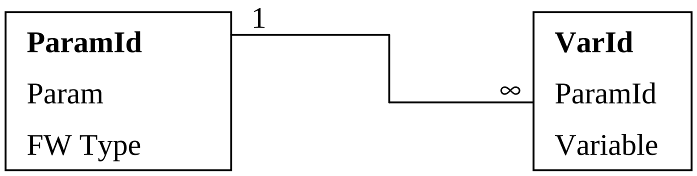


Figure 4.5: Tables Parameter & Variable

It is important that this solution suits the configuration syntax for both Watchguard and NetScreen and also other firewalls such as Cisc o PIX. The application should be as easy as possible to upgrade to handle other firewalls as well. The syntax must always be, as described above, a parameter followed by one or more variables separated by spaces.

For every firewall type, a typical relation ship between parameters and variables should be added by the database administrator using the two tables in Figure 4.5. The parameters, variables and their relationship can be modified anytime. A modification affects the configuration file. This solves the problem with versions updates.

The next problem involves the fact that not all firewalls follow the predefined parameter to variables pattern. Some of the firewalls need variables that are not defined in this pattern. For these firewalls we need to be able to replace the variables in the pattern with specific ones. To sort this out we added a new table to the parameter-variable structure (Figure 4.6).

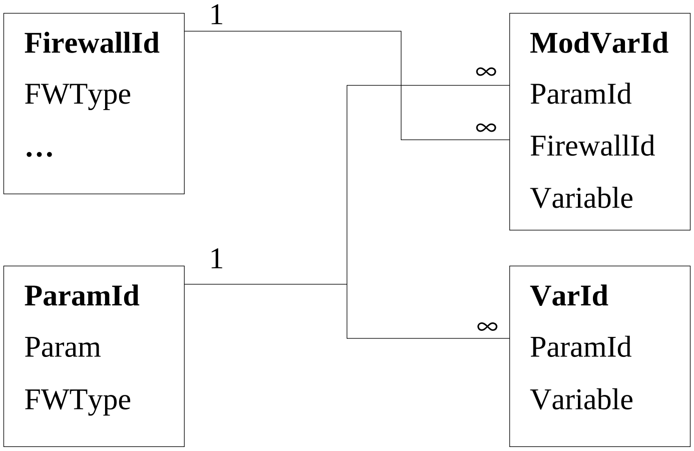


Figure 4.6: Tables ModifiedVar, Parameter & Variable

The new table, ModifiedVar, is similar to table Variable part from the attribute FirewallId. This is needed to relate to a certain firewall. With this table it is possible to delete or add new variables related to any parameter for a specific firewall. For example: a parameter called p1 has, according to the pattern, a variable v1. This command line in the configuration file would look like “p1 v1”. If the concerned firewall, f1, needs variable s v2 and v3 instead of v1, these two variables are simply inserted into the ModifiedVar table along with the parameter id for p1 and the firewall id for f1. Now, to build a configuration file for a firewall that does not follow the pattern, all variables t hat exist in the ModifyVar table, for that firewall id, should replace the variables in the Variable table where the parameter id attribute agrees with the parameter id attribute in ModifiedVar. This is either done in the application or it is done in the database using so -called Stored Procedure, which is an operation that is stored with the database server. Stored procedures are mostly written in SQL.

There exist both advantages and disadvantages with these two methods. When it is handled within the appl ication, it is more reliable than if the application is to depend on the database server, with the stored procedure, to function properly. Using stored procedure it will run faster because all information can be processed locally. It may also be easier to implement. The question is if this quicker solution is really necessary. Most of the configuration files follow the parameter -variable pattern, which means that this problem will not often occur. Because of this, and that no external source should be able to cause the application to generate non -valid configuration files, a decision was made to handle this within the application.

The complete database structure is located in appendix C.

4.2 Application types, Dialogs, SDI:s & MDI:s

There are three types of applications in Windows. There are dialogs, single document interfaces (SDI) and multiple document interfaces (MDI).

Dialogs are the simplest types of windows. These windows consist of only one class and cannot contain any other frame window. The class name usually ends with the letters Dlg. This makes it easier for the programmer to know that the class represents a dialog.

The Firewall Configuration System application is a SDI application. SDI:s allow one open document frame window to be opened in the mainframe window. These types of applications consists of three classes:

- The Frame class
- The Doc class
- The View Class

See section Classes 4.4 for more details.

MDI applications allow multiple document frame windows to be open in the same instance of an application. An MDI application has a window within which multiple MDI child windows, which are frame windows themselves, can be opened, each containing a separate document.

4.2.1 Why is this project based on a SDI application?

The FCS application might as well be dialog based. But since the whole user interface consists of 11 windows, a SDI based main window felt appropriate as a base for the design structure.

4.3 The Graphical User Interface

As the database structure was completed we could move on to the user interface. To be able to log users, a login window must be the first to appear when the application is executed (see Figure 4.7). On the login window there are fields for the user to apply his/hers userna me and password. In case the user fails to login or connect to the database, the cancel button will exit the application.

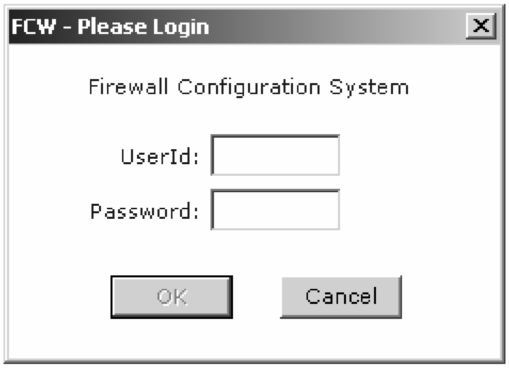
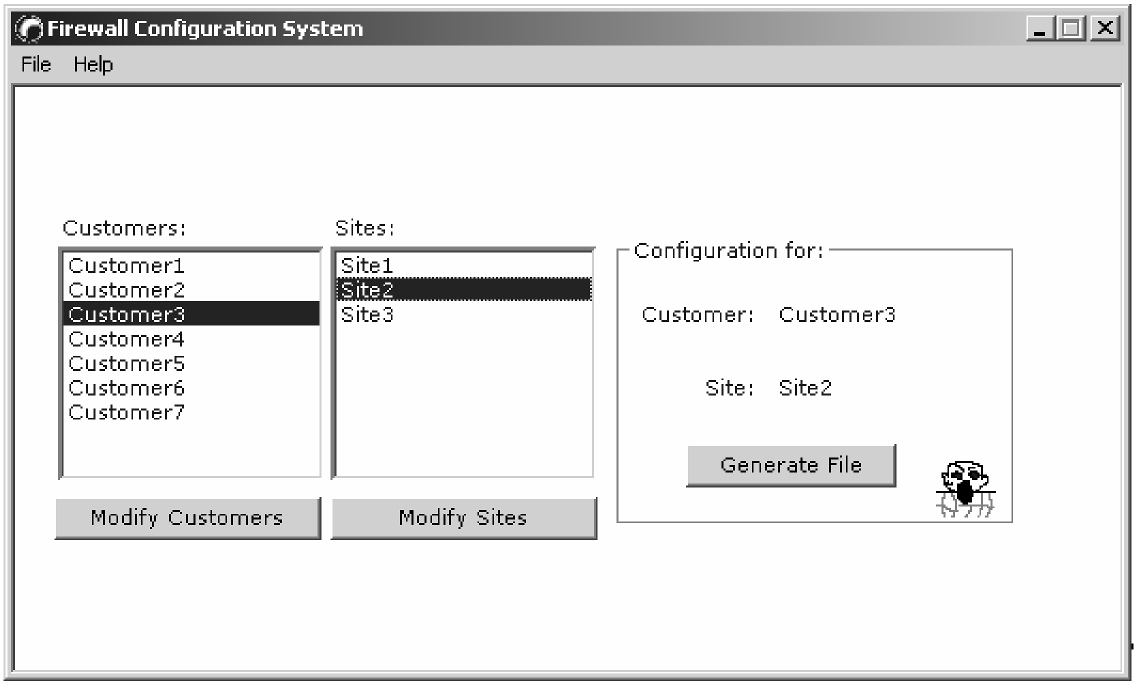


Figure 4.7: Login Dialog

When the user has logged in the main window (Figure 4.8) will appear and the login window will terminate. Here there are two list boxes, one for Customers and one for the customer's Sites. A list box is a window with a fixed size that provides a list of items to choose from. If the number o f items exceeds the number that can be displayed, a scroll bar is automatically added to the list box. The customers' names are listed in the Customers -list box as soon as the window is loaded. When selecting a customer its sites will be listed in the Sites-list box. The Modify Customers and Modify Sites buttons are used when the user wants to modify either a customer or a firewall. The Modify Sites button becomes enabled as soon as a customer has been selected. The Generate File button will allow the user to generate a configuration file for the selected customer and site. As soon as a customer and a site have been selected, they will appear above the Generate File button, which will become enabled.



Customer: Customer3

Site: Site2

Figure 4.8: Main SDI Window

When the Modify Customers button is pressed a new dialog appears with the main window still in the back (Figure 4.9). To add a customer the user will apply the name of the customer in the field next to the Add but ton. When this is done the Add button becomes enabled so the user can press it to add the customer. When the user wants to rename a customer he/she selects the customer and presses the Rename button, which is now enabled.

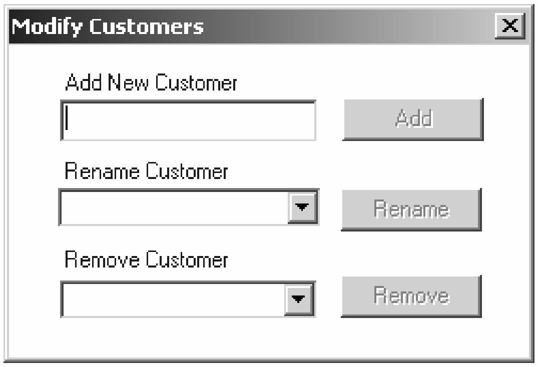
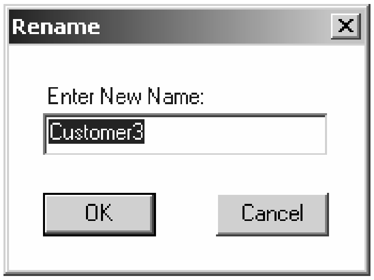


Figure 4.9: Modify Customers Dialog

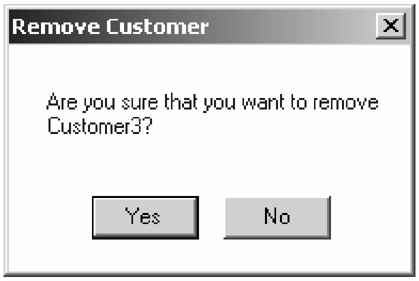
This will open the dialog in Figure 4.10 with the selected name is shown in a field. This is done so the user is sure that he/she is renaming the right customer. The user deletes the name and writes in the new customer name and presses the OK button. The new name will be saved and the window terminates. If the user does not want to rename the selected customer he/she presses the Cancel button. The window will terminate without any changes taking place.



Customer3

Figure 4.10: Rename Customer Dialog

If the user wants to remove a customer he/she selects the customer. This will enable the Remove button. As the user presses the Remove button and a new window appears (Figure 4.11).



Are you sure that you want to remove Customer3?

Figure 4.11: Confirm Dialog

Now back to the main window. If the Modify Sites button is pressed, a new window will appear that is quite similar to the Modify Customers window (Figure 4.12). The only difference is that the Rename button here is called Edit.

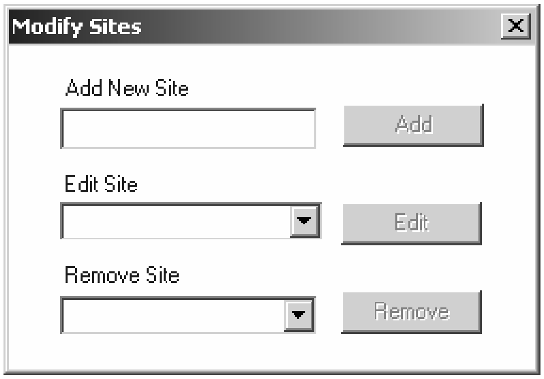
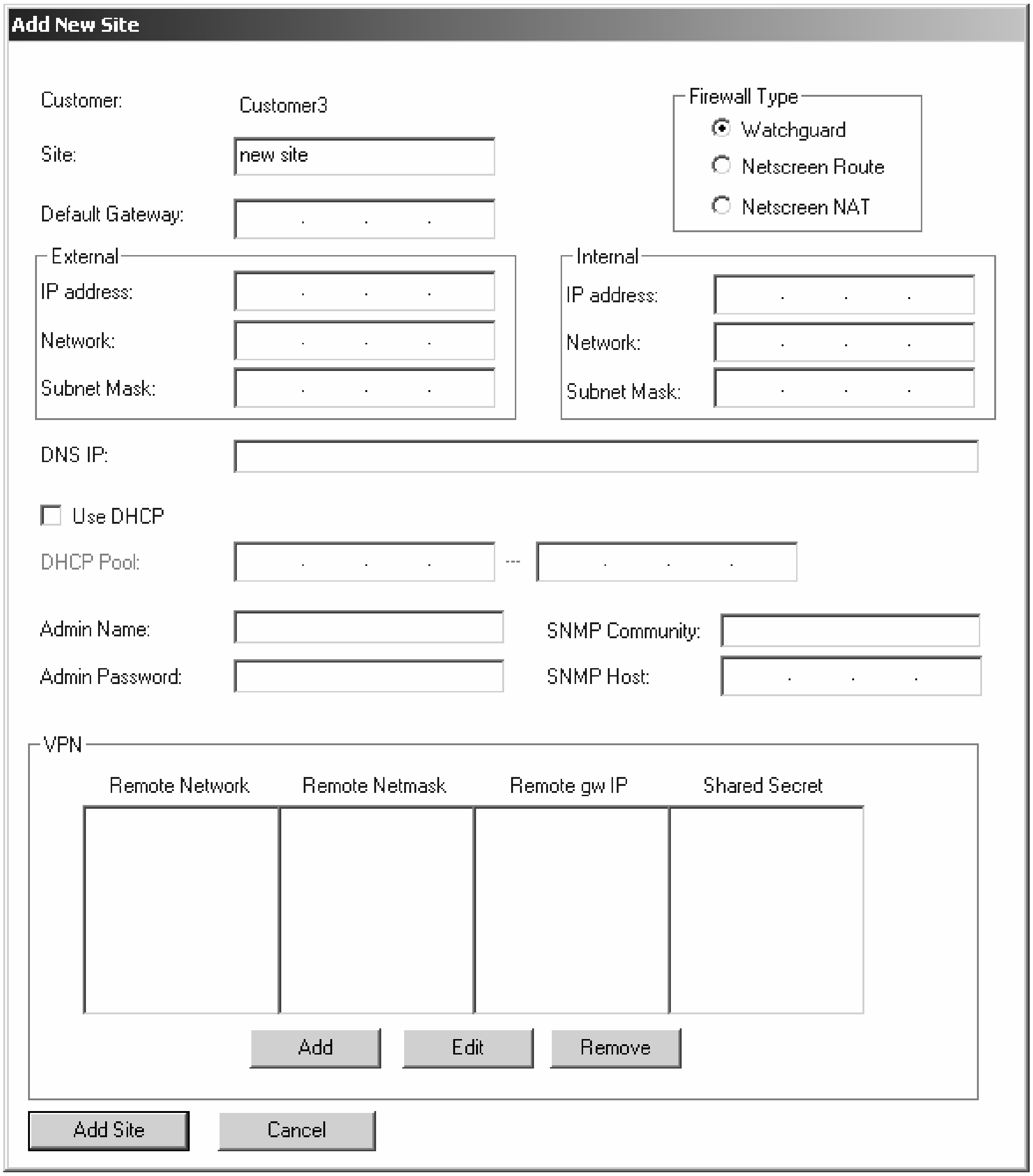


Figure 4.12: Modify Sites Dialog

If the user wishes to add a firewall he/she simpl y applies the name in the field next to the Add button. This button will then be enabled and by pressing it a new window will appear. We will call this window for “Add/Edit firewall dialog” since it is used both for adding and editing a firewall. The only difference is that the window title and the name of the button located down in the left corner will change depending on if the Add or Edit button was pressed.

When editing a firewall, the user selects the firewall that he/she wishes to edit. This will enable the Edit button, which when pressed opens the Add/Edit firewall dialog (Figure 4.13).



Add New Site

Customer: Customer3

Site: new site

Firewall Type
● Watchguard
○ Netscreen Route
○ Netscreen NAT

□ Use DHCP

Figure 4.13: Add New Site Dialog

When adding a new firewall this window will only contain the sel ected customer and the firewall name that was entered before pressing the Add button. In this window the user applies information that is specific for this firewall. So first of all the user determines what kind of firewall this information is for by selec ting one of the radio buttons. If the user selects Watchguard the fields next to SNMP Community and SNMP Host will be disabled because this type of firewall does not need the above information. If the user selects either one of the NetScreen firewalls the SNMP Community and SNMP Host will be enabled because both firewall types use this information.

Then why have two different NetScreen types? Well, this is for later use when generating the configuration file.

At first only two types of firewalls where cons idered required, which means only two different parameter to variable patterns. Then it came to our knowledge that NetScreen -5XP can have two different types of configurations depending on if it where to serve as a NAT or a Route firewall. NAT (Network Add ress Translation) enables the internal network to use one set of IP addresses for internal traffic and a second set of addresses for external traffic. [7]

The easiest solution was to consider the NetScreen-5XP as two different firewalls with two different patterns. Then by allowing the user to select firewall type the application knows which parameter to variable pattern to use when generating the configuration file.

The check box Use DHCP is optional for all the firewall types. If checked, the two IP address fields to the right of DHCP Pool will be enabled so the user can apply data. For the VPN information there is a list box with four columns. The columns are Remote Network, Remote Net mask, Remote gw IP and Shared Secret. When th e Add or the Edit button is pressed a new dialog appears (Figure 4.14). This window is called Add/Edit VPN. This window is the same for them both but the content in the data fields will differ. When Add is pressed and the list box is empty the Add/Edit VPN window will not contain any data.

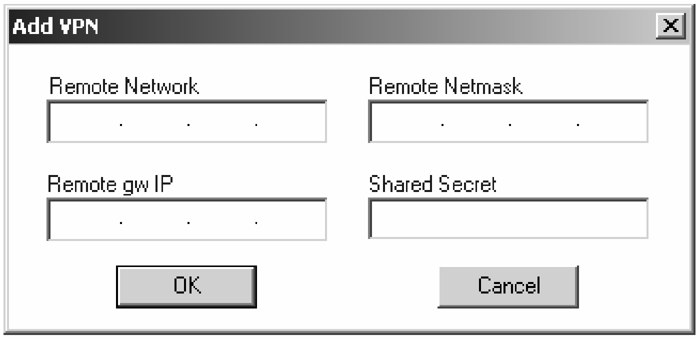


Figure 4.14: Add VPN Dialog

The window has three IP address fields, one for remote network, one for remote net mask and one for remote gateway IP. There is also a field for shared secret. When the OK button is pressed the window will terminate and the VPN information will be inserted into the list box in the VPN section of the Add/Edit firewall window. If data already exists in the list box of the VPN section and the user presses the Add button, the data from remote gateway IP and shared secret columns will be placed in the respective data fields in the Add/Edit VPN window. This is done because different Remote Network and Remote Net mask addresses use the same Remote gw IP and Shared Secret information. If the user wants to change these he/she just need to apply the new data into the two fields.

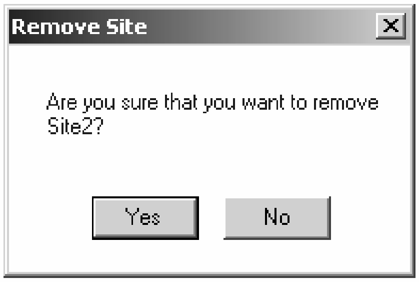
When editing the user selects a line in the VPN listbox that he/she wants to change and presses the Edit button. The Add/Edit VPN window will appear with and the data will be inserted into all of the fields. Overwrite the old data and apply the new and press the OK button. The window will terminate and the new data will overwrite the old in VPN section of the Add/Edit firewall window.

To remove VPN information the user selects the Remote Network address and presses the Remove button.

When the user is done applying data in the Add/Edit firewall window he/she presses the Add Site button to close the window and store the new firewall.

Now if the user presses the Edit button the Add/Edit firewall window will appear with all of the data that exists for the selected firewall inserted into the right data fields in this window. The user makes the changes that he/she desires and presses the Apply Changes button to terminate the window and save the changes.

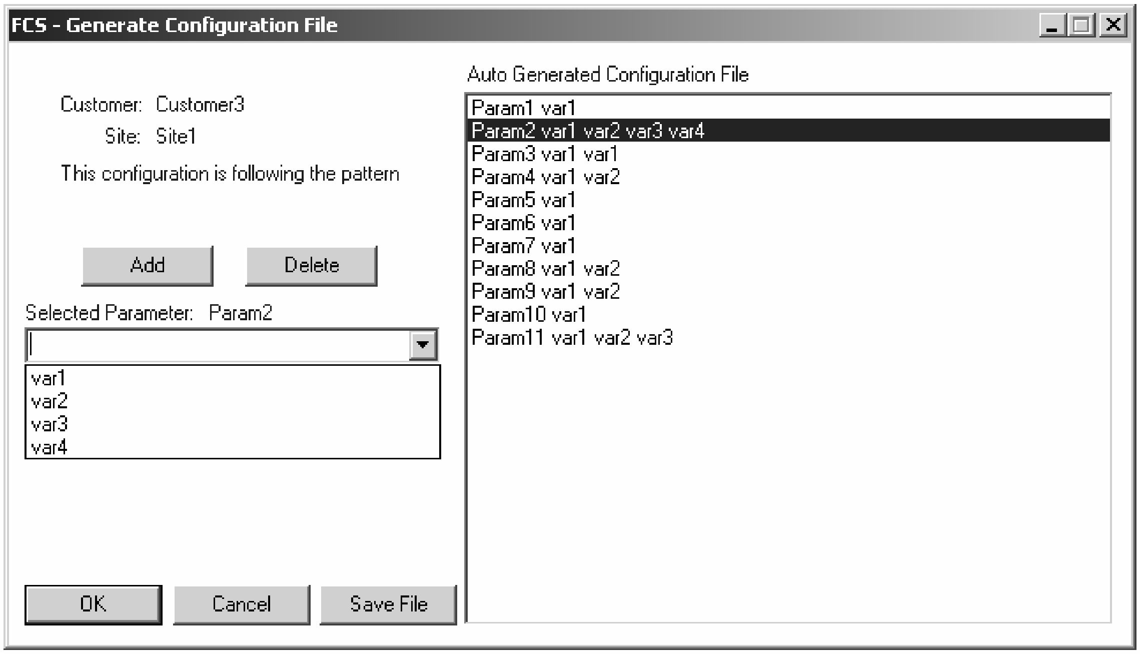
When the user wants to remove a firewall he/she selects the firewall, which will enable the Remove button. When pressed a new window is opened (Figure 4.15) that asks the user if he/she is sure that the selected firewall should be removed. If the user presses the OK button the window will terminate and the firewall will be removed. If the Cancel button is pressed the window terminates and no changes will take place.



Are you sure that you want to remove Site2?

Figure 4.15: Remove Site Dialog

Now back to the main window and the Generate File button. Pressing this button will show the dialog in Figure 4.16. This is the window that generates the configuration file. The name of the selected customer and site is shown so that the user knows what configuration file he/she is creating. It is also shown if the configuration is following the standard pattern for a configuration file or if there has been any modifications. The configuration file is placed in a list box as soon as the window appears. The user can make changes to it by selecting a certain command line from the list box. The parameter name of the selection will appear above the combo box that is now filled with the variables for that parameter. The user can now either add a new variable or delete an old one. To add a new variable the user applies the name in the field and presses the Add button. To remove a variable the user selects one from the combo box and presses the Delete button. If the user presses the OK button the window will terminate and the changes will be saved in the database for later use. On Cancel the window will terminate and no changes will take place. If the user is satisfied with the configuration file he/she presses the Save File button. This will save the changes to the database and a new window will appear. In this window the user applies the name of the file and where to save it.



FCS - Generate Configuration File

Customer: Customer3

Site: Site1

This configuration is following the pattern

Selected Parameter: Param2

```
var1
var2
var3
var4
```

Auto Generated Configuration File

```
Param1 var1
Param2 var1 var2 var3 var4
Param3 var1 var1
Param4 var1 var2
Param5 var1
Param6 var1
Param7 var1
Param8 var1 var2
Param9 var1 var2
Param10 var1
Param11 var1 var2 var3
```

Figure 4.16: Create File Dialog

4.4 Classes

This section is a brief description of all classes and their roles in the project.
All class names start with the letter ‘C’ to indicate their data type.

4.4.1 CAboutDlg

The GUI of the about box.

4.4.2 CDBThread

This thread handles all the database work. It is needed to prevent the main thread from waiting when connecting to the database etc, which causes the application to not respond.

4.4.3 CFCSApp

CFCSApp is the main thread of the application, which defines the class behaviors for the application. It creates and connects all components in the program including the main window user interface, which consists of CFCSMainFrame and CFCSView. This class receives all the event messages and passes them through to CFCSView.

4.4.4 CFCSView

Represents the area within the main window frame, the window that comes up when the user has logged in. The list boxes, buttons and other components that are within the main window are connected to this class. To interact with the database an instance of CDBThread is used.

4.4.5 CFCSDoc

The term document is referring to the data that is to be worked on in the program. This class is closely linked to the CFCSView class. It receives indata from the view class to process, the result is then sent back to the view-class for user display.

4.4.6 CLoginDlg

The GUI of the login dialog, which is shown before the main window appears.
Uses an instance of CDBThread to handle the login information.

4.4.7 CMainFrame

As the class name implies this class represents the window frame of the main window. The frame contains the menus, scrollbars and other visible objects that are connected to the window.

4.4.8 CMessageDlg

When the application wants to give the user a message, a box containing the massage appears to alert the user. This message box appears mostly to give error messages, for example when the database is down or when the login failed.

4.4.9 CSQLDirect & CSQLColumn

The direct interface to the ODBC database connection. These classes provide the functions needed to interact with the database.

4.4.10 CAddEditPortalDlg

If the user wants to add or change information for a selected firewall an object of this class is used.

4.4.11 CAddEditVPNDlg

This class handles the GUI of the add or edit VPN dialog. This dialog is shown when one of the two buttons, Add and Edit, are pressed in the CAddEditPortalDlg dialog.

4.4.12 CArrayEx

The CArrayEx is needed to create a 2-dimensional array. These arrays are used when fetching parameters and their related variables.

4.4.13 CCreateFileDlg

An object of this class is used to show the dialog where the creating of the configuration file is done.

4.4.14 CModCustDlg

For adding, removing or renaming a customer an object of this class is used to show the window where this is done.

4.4.15 CModPortalDlg

For adding, editing or removing a firewall an object of this class is used to show the window where this is done.

4.4.16 CNewCustNameDlg

When the Rename button is used from CModCustDlg an object of CNewCustNameDlg is used to show the window where the user can rename the selected customer.

4.4.17 CConfirmDlg

An obje ct of this class is used to show the little dialog that allows the user confirm that he/she really wants to delete an item.

4.4.18 CSaveFileDlg

When the user wants to save a configuration file an object of this class is used to show the window where the user can choose name and which folder to save to.

4.5 Class dependencies

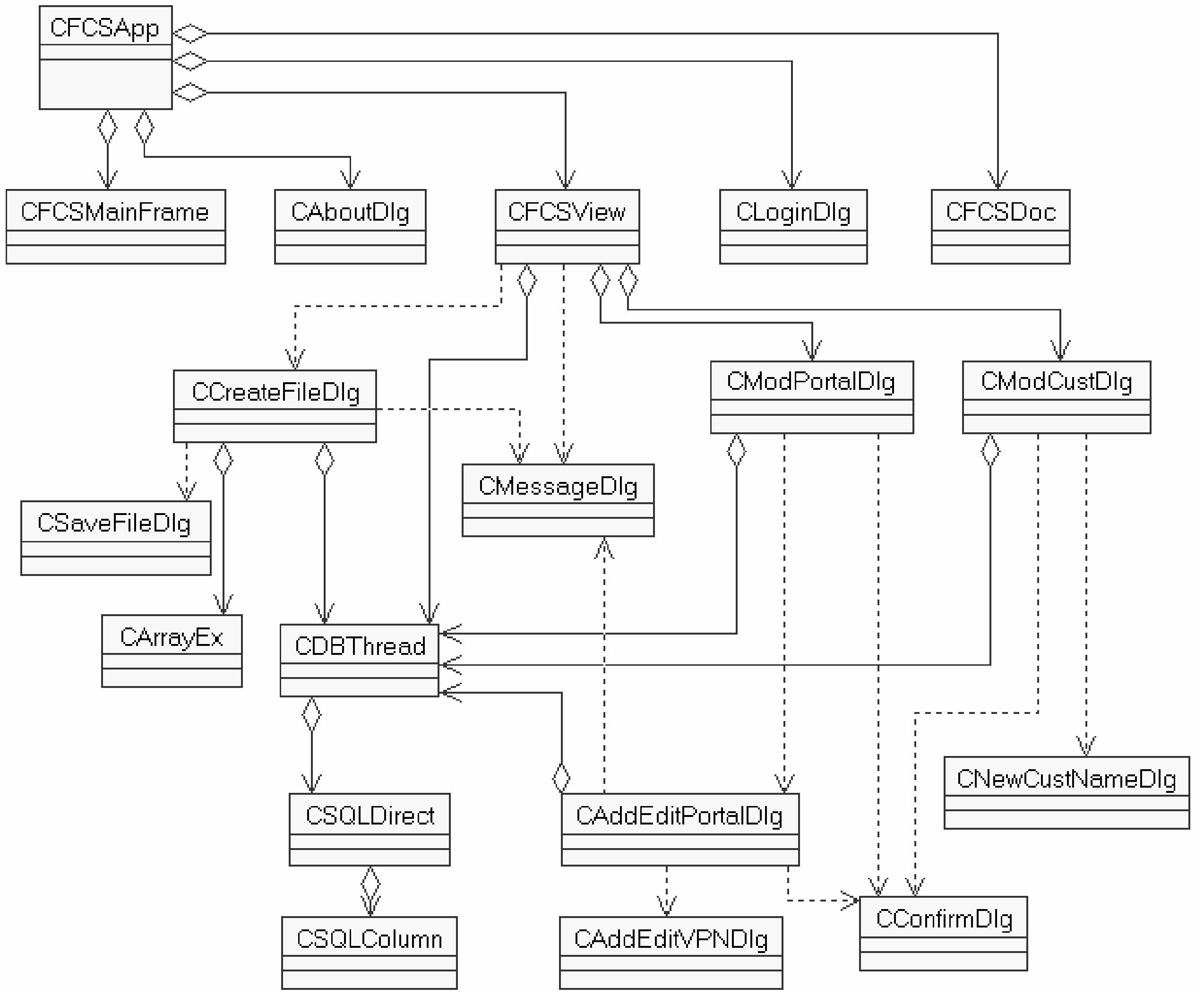


Figure 4.17: Class Diagram

5 Implementation

This chapter goes through the implementation along with the problems and solutions that we had to deal with during the development.

5.1 CDBThread – The thread that sends the SQL queries

This class is derived from CWinThread, which represents a thread of execution within an application. As you know by now, CDBThread handles all database interactions. This class was needed because if the main thread (the CFSCApp instance) calls the database by trying to connect or fetch data, this thread will get occupied while waiting. This prevents the application to inform the user about the situation. The window also stops to respond to user actions such as moving the window. The user might assume that the application has stopped running. With the help of this extra thread this problem is fixed.

For this to work, the main thread must know when the database thread is done with its task. Furthermore the database thread must know when to do what. To solve this messages are sent between the two threads, see Table 5.1.

Main thread = MT

Database thread = DBT

Messages	Description	Sender
WM_CONNECT	Request to connect to the database	MT
WM_ONCONNECT	Connection established	DBT
WM_ONCONNECTFAILED	Connection failed	DBT
WM_CLOSECONN	Request to close connection	MT
WM_ONCONNCLOSED	Connection closed	DBT
WM_CHECKLOGIN	Request to verify login information	MT
WM_LOGINOK	Login was successful	DBT
WM_LOGINFAILED	Login not approved	DBT
WM_SQLFAILED	SQL query failed	DBT
WM_GETCUSTOMERS	Request to retrieve all customers	MT
WM_FILLCUSTOMERLIST	All customers are collected	DBT
WM_GETPORTALS	Request to fetch all firewalls registered to a certain customer	MT
WM_FILLPORTALLIST	All firewalls for a certain customer are collected	DBT
WM_ADDCUSTOMER	Request to add customer	MT
WM_CUSTOMERADDED	Customer has been added	DBT
WM_RENAMECUSTOMER	Request to rename customer	MT
WM_CUSTOMERRENAMED	Customer has been renamed	DBT
WM_REMOVECUSTOMER	Request to remove customer	MT
WM_CUSTOMERREMOVED	Customer has been removed	DBT
WM_ADDEDITPORTAL	Request to add/edit a firewall	MT
WM_PORTAL_ADDED_EDITED	Firewall has been added/edited	DBT
WM_REMOVEPORTAL	Request to remove firewall	MT
WM_PORTALREMOVED	Firewall has been removed	DBT
WM_GETPORTALDATA	Request to receive firewall data	MT
WM_ONPORTALDATA	Firewall data has been received	DBT
WM_GETPARAMVAR	Request to receive parameters and their variables	MT
WM_ONPARAMVAR	Parameters and their variables are retrieved	DBT
WM_SAVEPARAMDATA	Request to store variables	MT
WM_ONSAVEPARAMDATA	Variables have been stored	DBT
WM_GETMODVARS	Request to receive modified variables	MT
WM_ONGETMODVARS	Modified variables have been received	DBT

Table 5.1: List of thread interaction messages

The application connects to the database server during authentication and then stays connected until the application terminates. That means that the CLoginDlg object must send the WM_CONNECT message to the CDBThread object. A problem that occurred at this point was that since the connection should last through the whole execution, the object of CDBThread could not be destroyed. If it is destroyed the connection is lost. So if the object is created in CLoginDlg it will be destroyed as soon as the user has logged in and the login dialog closes. Since the main window is not created until login ha s been granted the CDBThread object cannot be created in CFCSView either. To solve this a programming technique called *singleton* was used. Below is a simple way to implement this technique.

1. A static pointer to a CDBThread is created in CDBThread.

```
static CDBThread* _instance = NULL;
```

2. A static function that returns a pointer to CDBThread should be in the public section.

```
 static CDBThread* CDBThread::Instance()
{
   if (_instance == NULL)
  _instance =
   (CDBThread*)AfxBeginThread(RUNTIME_CLASS(CDBThread), NULL);

   return _instance;
}
```

An object of CDBThread is created the first time this function is called. The pointer to the object in 1) is then returned. Since the function is static it can be called before the object is created. The following occasions the function is called the pointer to the same object is returned, so only one object is created.

3. Now any object that wants to interact with the database should call the function 2) to retrieve a pointer. For example in CLoginDlg:

In the class definition:

```
private:
CDBThread* dbThread;
In the constructor:
```

```
//Get the static instance of CDBThread
dbThread = CDBThread::Instance();
```

To send messages to the CDBThread object the pointer returned from CDBThread::Instance() is used to address the message. Regardin g the opposite direction, that is, messages from the db thread to the main thread, a pointer to the executing object must be sent to the db thread before any interaction has begun. For this purpose two functions where added to CDBThread; SetParentDlg and S etParentView. Which of these functions that should be called depends on the type of object that uses the thread. SetParentView must be called before the main SDI window starts to request data from the database. All the other windows are dialogs so they wil l call SetParentDlg. The address of the calling object is sent as a parameter to the function. For example in CLoginDlg:

```
In the constructor:
//Get the static instance of CDBThread
dbThread = CDBThread::Instance();
//Send this to the thread so it can send messages here,
//do not send messages to dbthread before this call
dbThread->SetParentDlg(this);

When the dialog is destroyed
//Let the dbthread know that this dialog is not using it
dbThread->SetParentDlg(NULL);
```

The goal was to make this class as genera l as possible so that all classes that wish to fetch or store data in the database may use an instance of it. That is why both SetParentDlg and SetParentView are needed so that both types of classes can use CDBThread.

CDBThread uses the CSQLDirect class to access the database.

5.2 CSQLDirect & CSQLColumn – The ODBC interface

CSQLDirect provides the functions for interacting with a database via ODBC. CSQLColumn is a support class for CSQLDirect. These two classes were downloaded from the *Codeguru website* [4].

5.3 CLoginDlg – The Login Dialog

This class represents the login dialog window which is the first window that is shown when the application starts. When the application receives the username and password from the user these are sent forth to the database for verification. The login information is stored in an array of strings, which is sent as a parameter with the message WM_CHECKLOGIN. If the verification was successful CDBThread notifies this by sending WM_LOGINOK. If the verification fails the message WM_LOGINFAILED is sent. The message sequence when a user has logged in successfully is shown in Figure 5.1.

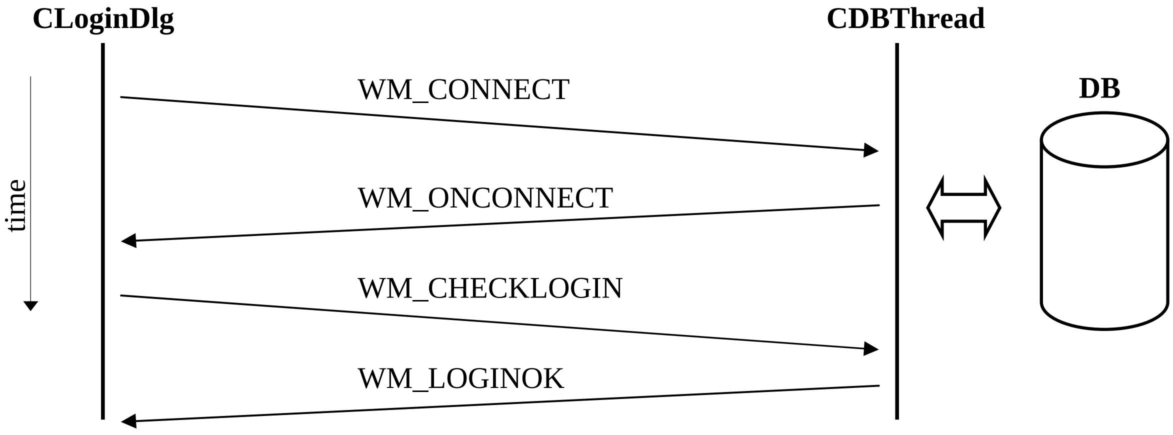


Figure 5.1: Scenario when login is successful

If the user successfully passes the authentication, the username is stored in the CDBThread instance for later logging. Every time a user adds or modifies a customer/firewall the thread logs the event. Even now, as the user is authenticated, this event is stored in the Log table.

To show the login dialog a derived function named DoModal() must be called. This function return either IDCANCEL or IDOK depending on the button pressed to close the dialog. If the user fails to login IDOK will not be returned, even though the OK button was pressed to confirm the authorization. When login fails the dialog should not be closed, only the cancel button will close the dialog if the login information is not approved. This means that any object that wants to show the login dialog knows if the user has logged in correctly or cancelled by checking the return value of DoModal(). Normally, this function is only used once, which is described in the following section.

5.4 CFCSApp – The main thread

This is the starting point of the execution. When the global variable, theApp, which is an instance of CFCSApp, is created, the main thread of the application is born.

It is in the function InitInstance() (see appendix D) that the main SDI window is built up and shown. Keep in mind that the login dialog needs to be showed before the SDI to grant users access to the application, so this also has to be done in InitInstance().

5.5 CMainFrame – The main window frame

CMainFrame is derived from the CFrameWnd class that encapsulates the functionality of a Windows single document interface (SDI) frame window. A frame window is a window that frames an application. If something needs to be changed concerning the frames behavior or how it looks this is implemented in CMainFrame.

In the function PreCreateWindow(…) (see appendix D) the maximize button is inactivated and the frames size in width and height is initialized. The frames smallest and largest height is set to the initiated height so the user cannot change this when the application is running. The same is done for the width. This is done in the function OnGetMinMaxInfo(…) (see appendix D).

5.6 CFCSView – The main window view

As this class represents the graphic area within the main SDI frame, the two list boxes (see chapter 4.3) there are components of this class. One of them is filled with all customers registered in the database and the other shows all firewalls related to the selected customer. When the main window appears the customer list is automatically filled, which is done by sending the WM_GETCUSTOMERS message to the CDBThread object. All customers are stored in a dynamic array that holds CString objects, which are dynamic strings. This array is created in the CFCSView class definition and only the reference to this array is sent to the CDBThread object. This means that both threads work with the same array. The database interacted thread fills the array with all registered customers and sends a message back to the main thread when all customers are fetched. On this message the main thread fills the customer list box using the array. This method, which is used to fetch multiple data from the database, is applied through the whole application. So a similar method is used when the user selects one customer from the list box and the application needs the fetch all related firewalls. Figure 5.2 servers as a demonstration.

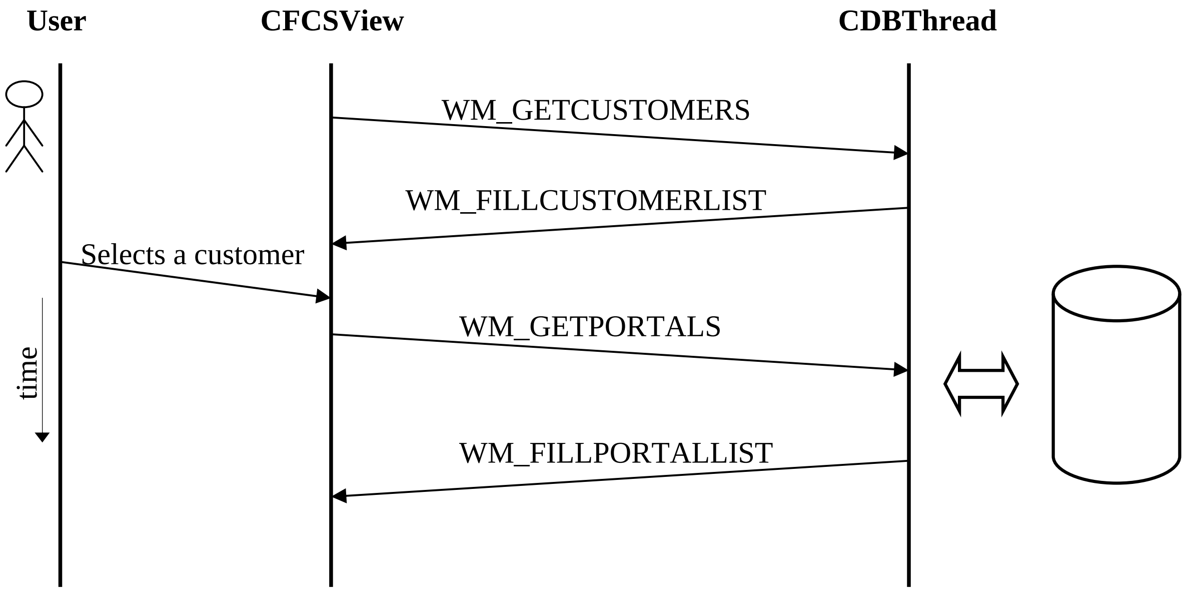


Figure 5.2: Successful selection of getting all firewalls related to the selected customer

This is pretty much what this class does except being the application base by providing buttons that open the dialogs that build up the application.

5.7 CAddEditPortalDlg – The dialog for adding/editing a firewall

The big dialog window showed in Figure 4.13 on page 21, is an instance of this class. It is used when a new firewall is registered and also when an existing firewall is edited. Depending on which, the properties, such as the window title, of the dialog changes. There is a private member variable in this class that is named ’addPortal’. It is of type boolean and is used as a reminder, telling the obj ect if the user pressed the add button or the edit button to launch dialog. This information is useful because if the user pressed edit in the Modify Sites dialog (Figure 4.12), all information stored associated with the selected firewa ll should be fetched from the database and displayed in the dialog that opens. An array to store firewall information is defined in the class declaration. This array is first filled with the customer name and the firewall name before the WM_GETPORTALDATA m essage is sent together with the array. This is information that the database thread must know in order to fetch the right data associated with the firewall. As described in section 5.6, a copy of this array is not used in CDBThread, only the reference.

For the application to properly extract data from the array, the order in which the database thread adds the data must be defined. The flow chart in Figure 5.3 clarifies this order.

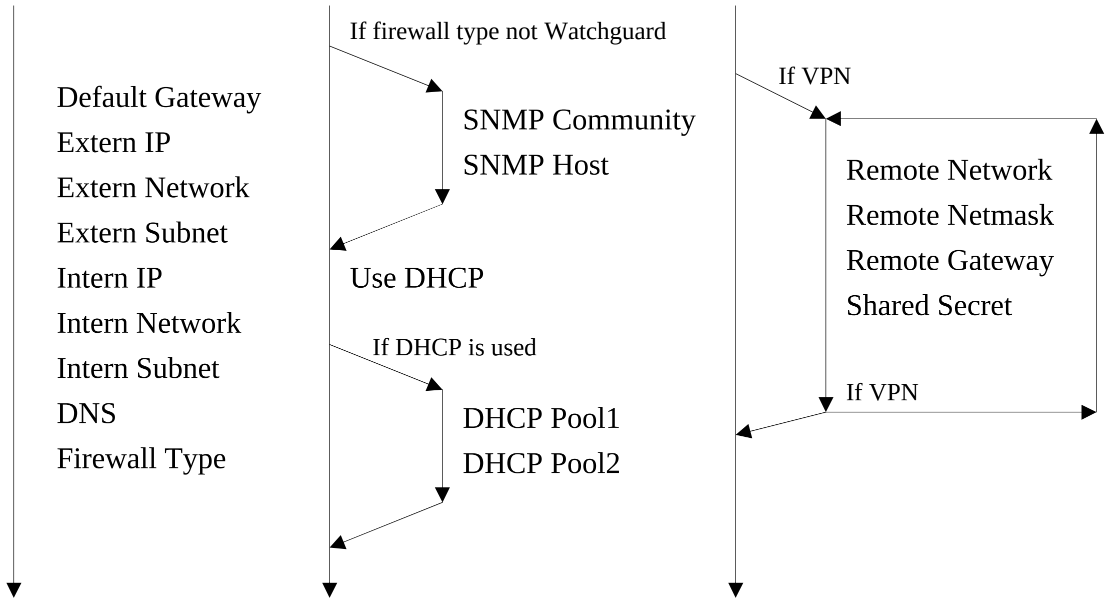


Figure 5.3: The order in witch firewall data is fetched

The array is cleaned in the database thread after that the customer name and firewall name are stored. Then the address of the default gateway is added to the array followed by the extern IP address according to the figure. Since firewalls of type Watchguard SOHO do not use the SNMP attributes, these are not added to the array if the selected firewall is a Watchguard. The progress is similar concerning DHCP. If the checkbox for DHCP (see Figure 4.13) is checked, the two DHCP attributes are added to the array. VPN information is grouped into four attributes. If one of them exists, all four must be fetched. As described in section 4.1.2, one firewall is not limited to just one VPN group. The database thread adds every group it can find related to the firewall to the array.

At this time the array is filled with all information needed. When the application reads the array the principle of Figure 5.3 is known.

If the add button in the Modify Sites dialog is pressed, a new firewall will be created. In this case no data should be fetched. All fields in the AddEdit dialog should be empty for the user to fill.

There is not a big difference between the add and edit dialogs regarding the procedure that handles the new written data to be saved. The only difference is the data that the application puts into the array before sending it to the database thread. In this case the variable addPortal is added to the array, which will tell the database thread to create a new row in the firewall table or just update an existing row. The customer name and firewall name are, as when fetching data, also added to the array. This firewall name is the name chosen before pressing the add/edit button (see Figure 4.12). One more thing is added before sending a message to the database, which is the text value in the site field. This value changing means that the firewall is r enamed. When the database thread updates a firewall that has changed name, the thread first uses the customer name together with the old firewall name (unique) to find the right firewall id. Thereafter the old firewall name in the database is updated with the new. In the case of adding a new firewall the old firewall name is not used.

The remaining fields in the add/edit dialog are added to the array using the principle in Figure 5.3. When the array is filled, WM_ADDEDITPORTAL is sent to the database thread along with the reference to the array.

When the user has added or edited a firewall this must be logged. Since the instance of CDBThread was given the username during the authorisation, the current user is stored in the log table alon g with the firewall id, date and event, which is either add a firewall or edit a firewall.

5.8 CCreateFileDlg – Where the configuration file is built up

When the user clicks on the “Generate File” button located on the main window a dialog is shown which is an instance of CCreateFileDlg. Here the user preliminary can examine the configuration file associated with the currently selected firewall. Modifications that break the parameter - variable pattern can also be made here. When this dialog window is initiated a list box is automatically filled with all parameters and related variables associated with the firewall concerned. With the help of this list box, the user has a chance to check the configuration file before it is written to a file.

There are two ways to fill this list box depending on if the pattern is followed or not.

5.8.1 Building configuration files based on the pattern

There are three tables to consider here:

- The Firewall table (Figure 4.3),
- The parameter table (Figure 4.5) and
- The variable table (Figure 4.5).

To begin with, all information regarding the firewall must be fetched. This is specific information, used only with the selected firewall, which will appear as variables in the configuration file.

For example; DefGateway 121.121.121.121.

The variable value associated with the DefGateway: parameter varies depending on what was written in the AddEdit dialog (see **4.3**. GUI). When the database admini strator defines a pattern for a firewall type, like Watchguard SOHO, aliases are used in the variable table to make the configuration file depending on the firewall. An example: The variable for parameter P should be the Administrator Name of the firewall. Writing ”AdminName” in the table Variable on the same row as the parameter id for P does this. When the application notices the text “AdminName” it is replaced in the configuration file with the real administrator name linked to the selected firewall.

The table below shows all aliases available when constructing patterns.

Alias
AdminName
AdminPassw
DefaultGateway
DHCP
DHCPPool1
DHCPPool2
ExtIp
ExtNetwork
ExtSubnet
IntIp
IntNetwork
IntSubnet
SNMPC
SNMPH

Table 5.2: Aliases used when defining standard configuration files

There is a minor problem regarding the alias solution. This occurs if for example the text "AdminName" is intended to be a variable and not a real administrator name . This matter is not regarded since it is most unlikely to occur.

To fetch the information stored for a certain firewall the same method as in CAddEditPortalDlg is used.

When all firewall specific data is received the parameters and related variables are still needed to construct the configuration file. The database thread will examine the tables Parameter and Variable to find the right pattern. It is possible now since the firewall type was fetched earlier. To fetch all parameters and related variables a dynamic array holding dynamic arrays is used. The parameters and variables are added to this array as Figure 5.4 shows.

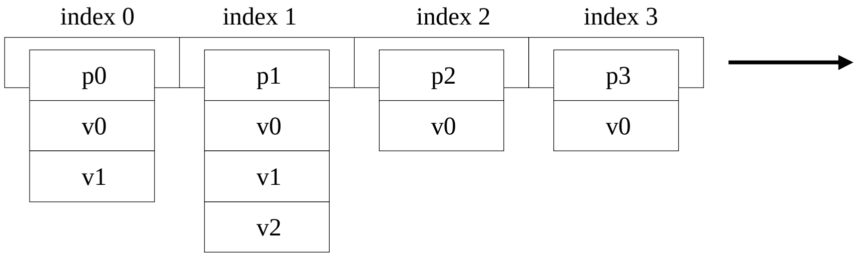


Figure 5.4: 2D array used to store parameters and variables

Items beginning with ‘p’ are parameters and whereas ‘v’ items are variables. To build the first command line, the array at index 0 is extracted. This array contains the first parameter along with its related variables. The contents are then added to the first line in the list box separated by spaces. The remaining command lines are filled to the list box using the same procedure. When the entire array is extracted the whole configuration file is available to review in the list box.

When the configuration file is stored, the whole list box content is written to the file.

5.8.2 Building modified configuration files

In order to deal with configuration files that do not follow any pattern, the solution described above must be slightly extended. There is one more table to be aware of in addition to the three tables used before. The ModifiedVar table was intended to be the answer to special firewall configurations.

As already explained, some firewalls may not use the predefined pattern for its type. New variables can be used and old ones can be deleted, but the parameters are always the same for each type. If the parameters change, a new firewall type, with a new pattern, has to be introduced. This is what has been done with NetScreen -5XP. This firewall has been considered as two different types, Nat and Route (see chapter 4.3).

The ModifiedVar table contains three important attributes:

- Variable

  Here the variable is stored.
- ParameterId

  Used to attach the variables to the right parameter.
- FirewallId

  Used to know witch firewall is using these extra variables.

Apart from the ModifiedVar table, two more arrays are needed. To simplify explanation, let us name the three arrays. Since the array described in 5.8.1 is still usable, this will be referred to as A. One of the new arrays has the exact same structure and types as A, refer to this one as B. The third array is a dynamic array of integers, C.

Let us consider scenario when the user opens the CreateFile dialog (Figure 4.16). The selected firewall is currently using the standard parameter pattern for its type. When the dialog opens array A is filled as described in 5.8.1, but not written to the list box. Instead it is copied to array B, from which data is extracted to fill the list box. This will soon make sense.

When the user selects a line in the list box, all variables there will be loaded into the combo box. By knowing which line that is selected the related variables are fetched from array A and B by using the line number as an index. The command at line N in the list box can always be fetched from array B at index N. As described above, this array at index N will contain the parameter at the first position, followed by all variables. Only the variables are loaded into the combo box since they are the only ones allowed to edit.

Let us say the user selects a line in the list box and adds a new variable by writing a new name in the combo box and pressing the Add button. The new variable is added to the array in array B at the same index as the line selected. Array A is not modified, the standard pattern is still stored there. That is the intention of A: s existence, to always hold the standard pattern for the firewall configured. If the modified command line now consists of parameter p0 followed by variables v0, v1 and v2, where v2 is the newly defined variable, arrays A and B would look like described in Figure 5.5.

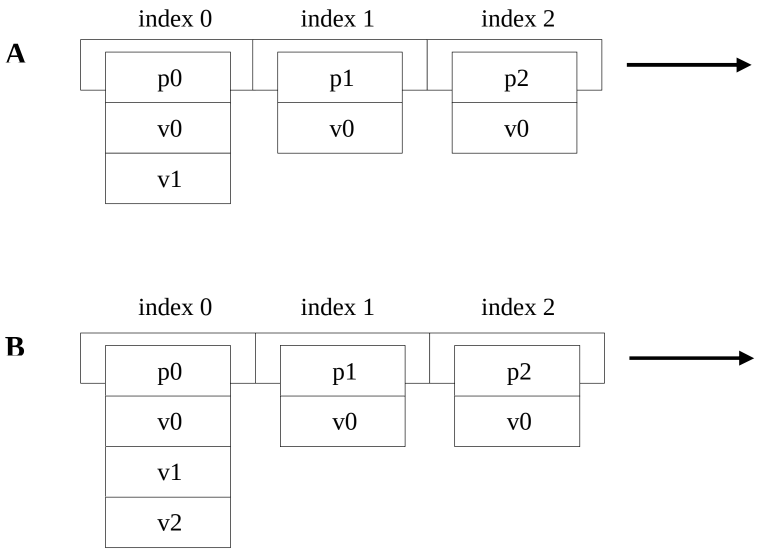


Figure 5.5: Arrays A and B when a new variable is added

Now the list box updates by using the array at index 0 from B. Next time the user selects this line from the list box, the combo box will also contain the new variable v2. To fill the combo box the variables are not simply extracted from array B at index 0 as when the list box is updated. Both arrays are used to fill the combo box. First all variables found in A at pos 0 are added. Then all variables in B at pos 0 that are not in A at pos 0 are added. This means that v0 and v1 are extracted from A and v2 from B.

The following will explain why A always contain the standard pattern and why the combo box is filled using both A and B:

After v2 is added and the selected line is still the top line, the user selects v1 from the combo box and clicks on Delete. This will delete v1 from array B and A is still unchanged (Figure 5.6).

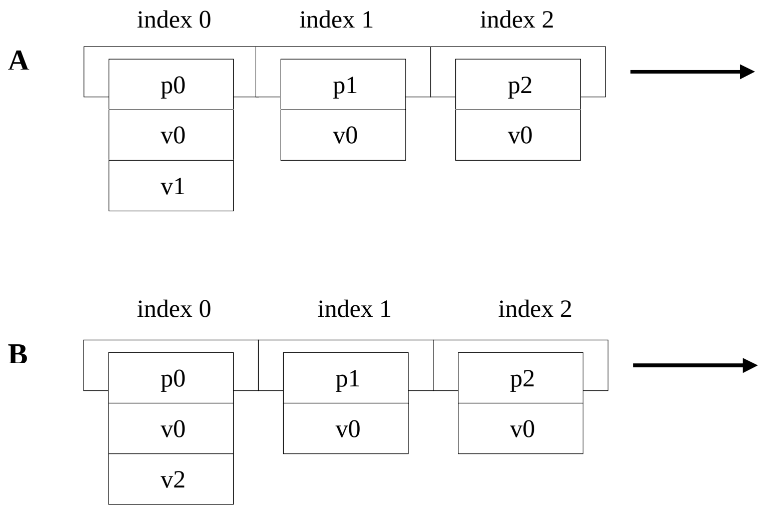


Figure 5.6: Arrays A and B when a variable is deleted

By using the same methods as described above, the list box and combo box will be filled as shown in Figure 5.7.

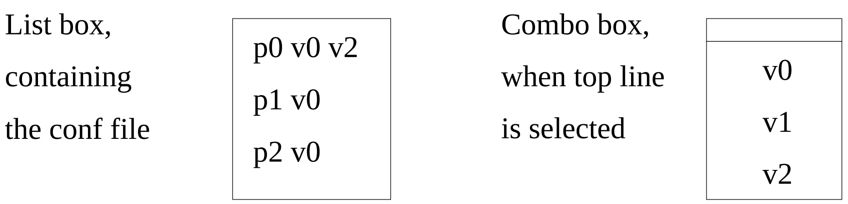


Figure 5.7: List box and combo box when a standard variable is deleted

The variables that were defined in the pattern for this firewall type will always appear in the list box regardless if the user includes them or not. This is needed so that the user does not loose these variables that belong to the pattern.

The array C is used to keep track of all positions in B that do not follow the pattern. Every position in B that is not identical with the same position in A is added to C. In the example showed in Figure 5.6, the digit 0 will be added to C. Every time the user adds or deletes a variable, the resemblance between A and B is examined. If they are not identical for the position modified, the position is added to C, if it is not already there. A digit showing the position in C is removed only if A and B are identical at that position.

This means that if the user modifies lines 0 (top line), 2 and 4, C will contain these digits. If the command on line 4 is changed back to the pattern, ‘4’ will be deleted from C.

As the user clicks on the OK button to save the modified configuration file, only the indexes found in C of array B will be saved to the ModifiedVar table. If C is empty, meaning that the firewall follows the parameter pattern, all rows in ModifiedVar with the matching firewall Id will be erased.

To open an already modified firewall configuration the arrays are built up to follow the method using arrays A, B and C:

1. A is filled with the parameter to variable pattern for the concerning firewall type.

2. All va riables found in ModifiedVar related to the firewall id are fetched from the database together with the correct parameters and stored in a temporary array D.

3. A is copied to B.

4. For parameters in B, which also exists in D, the whole array at that position is overwritten with the array in D. In other words, if variables for a certain parameter id exist in ModifiedVar, they are used instead of the pattern variables in table Variables.

5. Every time an array at a certain position in B is overwritten, that position is added to C.

By doing this every time the CreateFile dialog is initiated, the arrays are built up the correct way. A holds the standard configuration, B holds the real configuration and C remembers which variables related to a parameter that makes B differ from A. Keep in mind that if C is empty, A and B are identical, which means that the actual firewall is using the standard configuration.

6 Test & Evaluation

The purpose of this project was to facilitate the making of configuration files. This m eans that the application should be as easy as possible to set up and handle. If the application causes a lot of trouble it might as well be better to write the configuration files by hand.

The user should easily be able to understand how the application functions. Everything should be as smooth as possible.

- Special IP fields are used to inform the user that an IP address is intended. These fields only allow digits and the highest possible value is 255.255.255.255. This is good since the application should prevent human mistakes.
- When adding VPN information, Remote Gateway IP and Shared Secret are automatically filled when adding the second and continuous VPN data groups. Remote gateway IP and shared secret are usually the same for every network and net m ask (see Figure 4.14).
- To modify a configuration file is easy to achieve since the list box is showing the current file and the combo box contains variables to add or delete.
- It is not allowed to have two customers using the same name . Different firewalls are allowed the same name but not if they are registered to the same customer.

It is considered important to prevent users from irritation. Whether it is the colours, the location of controls or the way that the application is used. For example, the user should always be aware of what is happening. If the database server is very distant, the time interval needed to fetch data will increase. This will annoy the user if he/she is not informed about why there is a delay. Most messages di rected to the user will appear in the window title apart from error messages, which will popup in a modal dialog.

During interaction with the database most controls are disabled. The user must not press any button while the application is saving, removing or updating the database. There is no way for the user to interrupt these procedures. This means that if the database can cause the application to wait for data an undetermined time interval, which is uncertain, the application must be destroyed through a task manager. However, if the database will stop responding during for example a saving progress, there is no problem. The application will stop waiting and probably inform the user that something was wrong when trying to execute an SQL query.

As far as the database is concerned, it has to be accessed manually at some occasions:

1. When defining the standard configuration file for each type of firewall.
2. When the Log table is examined.

The first matter that has to be considered before the application will function is to prepare the database. All tables must be defined and then all parameter patterns must be written. When writing these patterns, the order in witch parameters and variables are stored in the tables are important. It is the same order that they will appear in the configuration file. The same applies for variables. For example: If the pattern says p0 v0 v1, and this line is modified in the application to p0 v1 v0, this command line is not following the pattern.

Secondly, an ODBC connection must be set up. This is needed for the application to be able to connect to the database. Here it is important that the defined ODBC name is known by the application, or else connection cannot be established. (This name is set in the file Messages.h before compilation)

No dynamic link libraries (DLL) were used. All classes built into the executable file. DLLs are needed to put classes in, which does not need to be loaded at start up. This speeds up the loading when the application starts. Since this application is not that extensive and loads fast, DLLs were not needed.

7 Summary & Conclusions

Internet Security Systems (ISS) gave this assignment to us and we are very glad to have accepted. It has been an interesting time for us to work on this application.

This project was started by discussing with ISS of how they wanted the application to work and what it would consist of. This gave us a good starting ground for the programming and setting up of the database. We wanted to start the design of the graphical user interface (GUI) but we soon realized that the best way to start was by constructing the database.

The database runs on MS SQL server to allow multiple clients to be able to access it simultaneously. It is structured per customer. This makes it easier to apply equipment and configuration for each customer.

The GUI is based on an SDI main window with ten dialog windows. The main window allows the user to modify customers/firewalls and to generate a configuration file for a specific firewall. During the actual coding of the project every window and its functions were constructed separately. This made it easy to test the application as it progressed.

The application communicates with the SQL Server database by connecting to an ODBC source, which means that an ODBC source must be defined for the application to function properly.

For the implementation we use two threads; a main thread and a database thread. If only one thread was to be used the application would not respond while working with the database. No information can be given to the user during this time. By using two threads this will not occur. Messages are sent between the two threads so they know when to react.

We are very satisfied with our work of building the application. There is a good code structure and not very complicated functions. There should not be any problems to search the code for errors. All text constants used in the GUI are easily changed in the file GUIConst.h, this can be desired to do for example when changing the language. But of course the project must be recompiled and rebuilt before any changes will take effect.

Regarding the CSQLDirect class some problems were encountered that we managed to bypass by coding in a different way. SQL queries that involved fetching data from multiple columns fails by returning data only from one column. This means that if data from three columns is needed, three different SQL queries have to be sent to the database, which will slow down the operation a bit. Here we are not satisfied with our solution. CSQLDirect should be repaired or replaced with another class that handles direct connection to the ODBC.

The application is still in lack of a few functions that will conclude the configuration files that are generated. No solution has yet been found regarding the VPN information that is entered for every individual firewall (Figure 4.14). Although this information is saved to the database, it is not a part of the configuration file. Since it is uncertain how many different VPN configurations that are needed for each firewall, a special method must be used to be able to include this information in the file. The solution must contain aliases in some way (Table 5.2). A similar problem th at is not implemented is the inbound and outbound traffic rules. These are very important since they tell the firewall what data to let through. At this time these commands will have to be applied to the generated configuration file manually.

List of Abbreviations

FCS – Firewall Configuration System

ISS – Internet Security Systems

VPN – Virtual Private Network

GUI – Graphical User Interface

ODBC – Open DataBase Connectivity

IPSec – Internet Protocol Security

AH – Authentication Header

ESP – Encapsulating Security Payload

DSL – Digital Subscriber Lines

MSS – Manage Security Services

ISDN – Integrated Services Digital Network

FTP – File Transfer Protocol

DES – Data Encryption Standard

CLI – Command Line Interface

SSH – Secure Shell

AES – Advanced Encryption Standard

DBMS – Database Management System

SQL – Structured Query Language

DNS – Domain Name System

SNMP – Simple Network Management Protocol

DHCP – Dynamic Host Configuration Protocol

SDI – Single Document Interfaces

MDI – Multiple Document Interfaces

NAT – Network Address Translation

DLL – Dynamic Link Library

References

[1] William Stallings. *Network Security Essentials*. Prentice Hall, 2000

[2] http://vpn.shmoo.com/

[3] MSDN Library (Visual C++ help), also on the net: http://msdn.microsoft.com/

[4] http://www.codeguru.com/mfc_database/direct_sql_with_odbc.shtml

[5] http://www.watchguard.com/products/wgls.asp

[6] http://www.netscreen.com/products/index.html

[7] http://www.webopedia.com

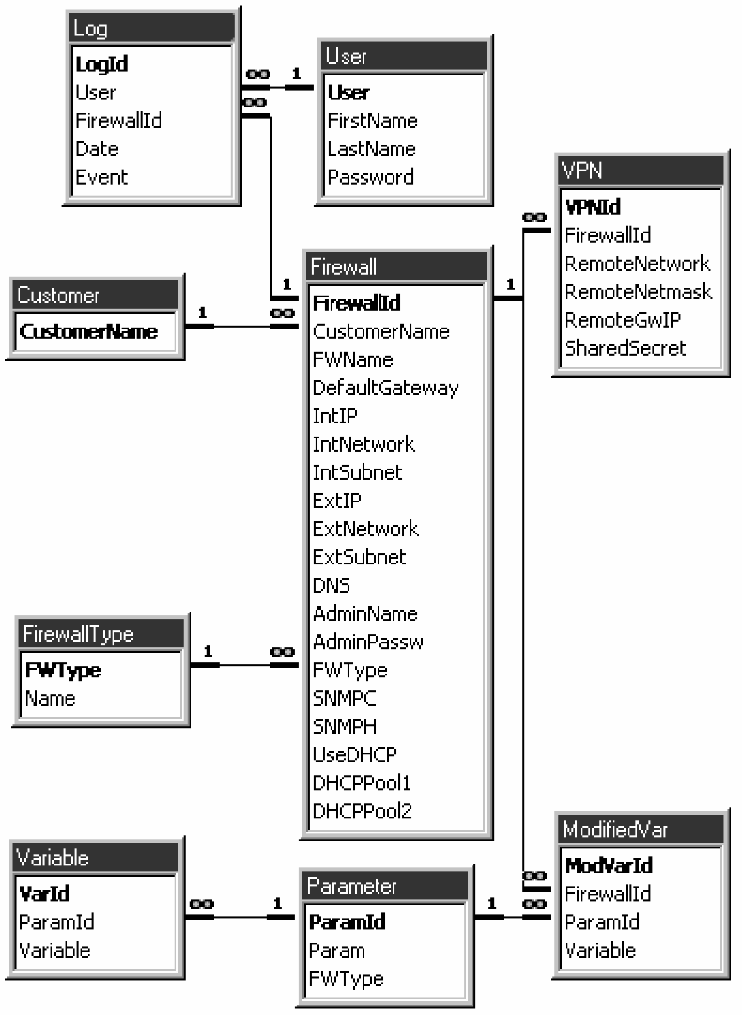
A Example of a Watchguard SOHO Config file

```
FDATE: Mar 22 2001
FTIME: 17:17:32
FVER: 2.3.16
config.platform: windows
config.version: 0.1
config.watchguard.dvcp.enable: 0
config.watchguard.id:
config.watchguard.modules: boot root ipsec proxy
config.watchguard.release: humptulips
config.watchguard.vendor: WGTI
config.watchguard.version: 4.00.B444
networking.bridge.external: 111.111.111.94
networking.dhcp_client.enable: 0
networking.dhcp_client.identifier: name
networking.dhcpd.default.default_lease_time: 86400
networking.dhcpd.default.default_rebind_time: 64800
networking.dhcpd.default.default_renew_time: 43200
networking.dhcpd.enable: 0
networking.dhcpd.firstip: 111.111.112.1
networking.ethernet.00: eth0 111.111.111.82 111.111.111.80 255.255.255.240 111.111.111.94
networking.ethernet.01: eth1 111.111.112.245 111.111.112.0 255.255.255.0 111.111.112.245
networking.ipsec.autostart: 1
networking.ipsec.enable: 1
networking.ipsec.policy.inbound.000.disposition: secure
networking.ipsec.policy.inbound.000.dst_ip: trusted
networking.ipsec.policy.inbound.000.src_ip: 111.111.112.0/24
networking.ipsec.policy.inbound.000.tunnelname: 000
networking.ipsec.policy.outbound.000.disposition: secure
networking.ipsec.policy.outbound.000.dst_ip: 111.111.112.0/24
networking.ipsec.policy.outbound.000.src_ip: trusted
networking.ipsec.policy.outbound.000.tunnelname: 000
networking.ipsec.remote_gw.SOHOGlobalGateway.id:
networking.ipsec.remote_gw.SOHOGlobalGateway.ip: 222.222.222.162
networking.ipsec.remote_gw.SOHOGlobalGateway.sharedkey: uElppGrLaTpNiDlter
networking.ipsec.remote_gw.SOHOGlobalGateway.type: isakmp
networking.ipsec.telecommuter.local_ip: 0.0.0.0
networking.ipsec.telecommuter.remote_ip: 0.0.0.0
networking.ipsec.tunnel.000.remote_gw: SOHOGlobalGateway
networking.ipsec.tunnel.000.sap.00.esp.alg: 1
networking.ipsec.tunnel.000.sap.00.esp.authalg: 1
networking.ipsec.tunnel.000.sap.00.life.kbytes: 0
networking.ipsec.tunnel.000.sap.00.life.seconds: 29030400
networking.ipsec.tunnel.000.sap.00.type: ESP
networking.ipsec.vpntype: SOHO
networking.nameservice.dhcpd.dns.0: 111.111.111.17
networking.nameservice.dhcpd.dns.1: 111.111.111.100
networking.nameservice.dhcpd.domain_suffix: somedomain.com
networking.nameservice.remote.dns.0: 0.0.0.0
networking.nameservice.remote.domain_suffix:
networking.nameservice.remote.wins.0: 0.0.0.0
networking.pppoe.enable: 0
networking.pppoe.idletimeout: 0
networking.pppoe.pass:
networking.pppoe.user:
options.admin.enable: 1
options.admin.name: sohogb
options.admin.pass: bsro13
options.controld.log_host: 212.212.212.254=34ff230bff401ffd0ffc1ff770ff5d04
options.controld.log_host.enable: 1
options.cskt.disable: 0
options.soho.feature_key: 54297BDD10648620
options.urltrack.enable: 0
```

B Example of a NetScreen config file

```
set clock ntp
set clock dst-off
set admin name "root"
set admin password nEHWJTrsXx6gcTlM4SCMrnPt5IMdGn
set admin manager-ip 222.222.222.0 255.255.255.0
set admin manager-ip 222.222.225.0 255.255.255.0
set admin sys-ip 0.0.0.0
set admin port 1212
set interface trust ip 0.0.0.0 255.255.0.0
set interface untrust ip 0.0.0.0 255.255.0.0
set interface untrust gateway 0.0.0.0
unset interface trust manage
set interface trust ping
set interface untrust manage ping
unset interface untrust manage telnet
set interface untrust manage scs
set interface untrust manage snmp
set interface untrust manage global
unset interface untrust manage global-pro
set interface untrust manage web
unset interface untrust ident-reset
unset interface untrust manage ssl
unset policy 0
set flow tcp-mss
set hostname firewall-name
set ntp server 192.5.41.40
set ntp interval 120
```

C The Database Structure



D Fragments of Source Code

CFCSApp – InitInstance()

```
BOOL CFCSApp::InitInstance()
{
 . . .
          //Show Login window
          if(!ControlLogin())
                    return FALSE;


          //SDI construction
          CSingleDocTemplate* pDocTemplate;
          pDocTemplate = new CSingleDocTemplate(
                    IDR_MAINFRAME,
                    RUNTIME_CLASS(CFCSDoc),
                    RUNTIME_CLASS(CMainFrame),
                    RUNTIME_CLASS(CFCSView));
          AddDocTemplate(pDocTemplate);


 . . .
// The one and only window has been initialized, so show  and update
it.
          m_pMainWnd->CenterWindow();
          m_pMainWnd->UpdateWindow();
          m_pMainWnd->SetWindowText(MAIN_WND_TITLE);
          m_pMainWnd->ShowWindow(SW_SHOW);

          delete loginDlg;
          loginDlg = NULL;

          return TRUE;
}
```

CFCSApp – ControlLogin()

```
BOOL CFCSApp::ControlLogin()
{
          int iResponse = loginDlg->DoModal();

          if(iResponse == IDCANCEL)
                    return FALSE;
          else if(iResponse == IDOK)
                    return TRUE;
}
```

CMainFrame – PreCreateWindow(…)

```
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
  ...
// TODO: Modify the Window class or styles here by modifying
//  the CREATESTRUCT cs

          cs.style &= ~WS_MAXIMIZEBOX;
          cs.cx = FRAMEWIDTH;
          cs.cy = FRAMEHEIGHT;

          return TRUE;
}
```

CMainFrame – OnGetMinMaxInfo(…)

```
void CMainFrame::OnGetMinMaxInfo(MINMAXINFO FAR* lpMMI)
{
          lpMMI->ptMinTrackSize.x = FRAMEWIDTH;
          lpMMI->ptMaxTrackSize.x = FRAMEWIDTH;
          lpMMI->ptMinTrackSize.y = FRAMEHEIGHT;
          lpMMI->ptMaxTrackSize.y = FRAMEHEIGHT;
          CFrameWnd::OnGetMinMaxInfo(lpMMI);
}
```

E Small Office Managed Firewall Service

Service Overview

Internet Security Systems’ (ISS) Small Office Managed Firewall Service is a customized solution specifically designed to meet the needs of the small business who, while having limited network access points on the Internet, still must concern themselves with ensuring they have taken the appropriate measures to minimize security exposures and limit unauthorized access, both inside and outside their enterprise.

This low cost service allows our customers to leverage Internet Security Systems’ security engineers for the configuration and ongoing support for their firewall, allowing their staff to focus on mission critical business priorities and projects. Through our Managed Firewall Service, Internet Security Systems, a renowned leader in the Internet security arena, becomes an extension to our customers’ staff.

Because the majority of firewall breaches are caused by the mis-configuration of firewall rules and properties, one key component of the Managed Firewall Service is the initial firewall setup process. Using ISS’ extensive experience and knowledge of security best practices, ISS’ security engineers have designed firewall configurations that will support our customers need for Internet access to maximize protection.

Service Detail

Platform Overview, Setup and Deployment

As a part of the service, the customer receives a certified firewall platform from ISS. The firewall platform is staged and pre-configured at ISS’ certified deployment center by a certified deployment engineer.

The Small Office (S0) Managed Firewall Platform and customer setup includes:

! One Certified Firewall Platform (including hardware, hardened OS, and software)

! Expert configuration of the firewall hardware

! * Selection and implementation of the most appropriate *SO* firewall configuration. The *SO* templates are:

1. Outbound-only Access Template: Allows all outbound Internet Access and VPN (if selected) traffic only. All inbound connection attempts to the customer network will be dropped.

2. Two-way Access Template: Allows all outbound Internet Access, allows VPN (if selected) traffic and allows inbound services to designated IP addresses. HTTP, HTTPS, FTP, SMTP, POP3, SSH, DNS, Telnet and/or one custom TCP or UDP port. All other connection attempts to the customer network will be dropped.

* This service is limited to supporting no more then 6 Internet accessible Servers/IP addresses per customer. Furthermore, depending on the chosen CPE, this service is limited to supporting one IP address per service.

! Remote management setup

The customer receives the firewall and follows a fully documented installation process, which primarily directs the customer through the process of connecting the firewall to their existing network. Partner/ISS phone support is available to the customer for assistance through this process. Because the firewall has been pre-configured, once it is installed ISS can immediately begin remote management of the firewall from our Security Operations Center. An encrypted Internet connection provides ISS' security engineers access to the firewall for remote maintenance of the firewall, including troubleshooting and problem resolution.

Ongoing Management

Once the firewall is remotely accessible by ISS, round-the-clock management of the platform commences.

If required, the customer has the opportunity to change their firewall configuration. A change can be defined as either: 1) a change from one *SO* firewall template to the alternate, Outbound-only to Two-way / Two-way to Outbound-only; or 2) within the Two-way Template the addition or change of up to three of the IP addresses or services included in the template. The customer can process up to four (4) Security Policy changes per year **. Customer change requests must be submitted to ISS via electronic submission method provided to the Partner. Internet Security Systems' security engineers will review and validate all customer security policy changes. Change validation and recommendations based on either technical issues or possible security compromises will be communicated back to the partner to initiate communication with the end customer.

The Small Office Managed Firewall Service includes:

! 24x7 monitoring and firewall management

! Ability to change firewall template, to meet customer needs, up to 4 times yearly**

! Timely platform upgrades, as deemed necessary by ISS for proper functioning

** The above stated rule-base constraints go into effect after the customers first 30 days of managed service. Customers exceeding 4 firewall rule-base changes per year will incur a $25 charge per additional rule-base change request. These charges will be billed annually on the customers contract anniversary date. Proactive rule-base changes made by ISS in the event of a security breach do not apply.

Service Level Agreements

Each new Customer is assigned a Deployment Engineer, (DE), who is responsible for the timely and successful implementation of the products and services purchased. During the turn-up process, the DE is Customer’s single point of contact regarding all issues.

The Service Levels are effective once all of the following have occurred:

(1) All outstanding issues have been resolved to DE’s satisfaction, including the successful installation and testing of, where applicable the required out of band access solution, and permanent software licenses on all managed security platforms.

(2) Once the implementation has been completed such that no outstanding issues exist, the support of Customer’s account is transitioned from the assigned DE to our Security Operations Center, which is available to assist with all questions or issues.

Rule-base Change Request Validation Guarantee (SLA 2)

Internet Security Systems’ Rule-base Change Request Validation Guarantee is to have a Internet Security Systems Security Engineer analyze each Rule-base Change Request that the Customer submits, and notify the Customer should any security risks be foreseen or additional information be required to allow for accurate implementation of the request. Validation is done to ensure that the change being requested is in the Customer’s best security interest, and follows best security practices. This validation will occur within four (4) hours of the receipt of the change request. If the Security Engineer determines the change request may cause a security risk or is lacking required information, the request will be placed in a “hold” status and the Customer’s validation communication from Internet Security Systems will state this status. Change Requests removed from “hold” status will be considered new change requests and treated accordingly. At Customer’s request, Internet Security Systems will determine the total number of Customer’s Rule-base Change Requests for a given calendar month that were not validated within the specified time frame. This guarantee is only available for rule-base change requests submitted by a valid Customer Security Contact in accordance with the Internet Security Systems Rule-base Change Request Submission Procedure. Customer is solely responsible for providing Internet Security Systems accurate and current contact information for Customer’s designated points of contact. Internet Security Systems will be relieved of its obligations under this guarantee if Internet Security Systems’ security contact information for Customer is out of date or inaccurate due to Customer’s action or omission. If Internet Security Systems fails to meet this guarantee the Customer’s account shall be credited the pro-rated charges for one day’s

Monthly Service Fee of the Customer's specific managed service, and if applicable specific managed security platform, related to the change submitted for which the rule-base change request validation guarantee has not been met. Under this SLA, Customer may obtain no more than one credit per contracted service per day. (VQ32001.9)

Rule-base Change Request Implementation Guarantee (SLA 3)

Internet Security Systems' Rule-base Change Request Implementation Guarantee is to implement Customer rule-base change requests within twelve (12) hours of ISS' receipt, unless the request has been placed in a "hold" status in the Validation process. This guarantee is based on actual time of implementation, and not on the time that Customer was notified that the request was completed. As set forth below, Internet Security Systems will credit Customer's account if Internet Security Systems fails to meet this guarantee during any given calendar month. At Customer's request, Internet Security Systems will determine the total number of Customer's Rule-base Change Implementation Requests for a given calendar month that were not implemented within twelve (12) hours. This guarantee is only available for rule-base change requests submitted by a valid Customer Security Contact in accordance with the Internet Security Systems Rule-base Change Request Submission Procedure. Internet Security Systems will promptly notify Customer upon implementation of a request by a method elected by Internet Security Systems (telephone, email, fax, pager, or electronic response via the MSS customer portal). Customer is solely responsible for providing Internet Security Systems accurate and current contact information for Customer's designated points of contact. Internet Security Systems will be relieved of it s obligations under this guarantee if Internet Security Systems' security contact information for Customer is out of date or inaccurate due to Customer's action or omission. If Internet Security Systems fails to meet this guarantee the Customer's account shall be credited the pro-rated charges for one day's Monthly Service Fee of the Customer's specific managed service, and if applicable specific managed security platform, related to the change submitted for which the rule-base change request implementation guarantee has not been met. Under this SLA, Customer may obtain no more than one credit per contracted service per day.

Small Office Managed Firewall - Customer Deployment Guarantee (SLA 9)

Internet Security Systems will make commercially reasonable efforts to ensure that Customer has a fully functioning managed firewall service available/deployed within three (3) business days of notification that all of the following have been completed:

1) ISS has received all of the information required from the cu stomer on the customer enrollment form

2) Customer has a valid usable static IP address for the firewall, and this information has been provided to ISS

3) Customer confirms it has successful Internet access

4) Customer has taken receipt of the firewall and completed successful implementation of the device, fully following the provided self-installation kit

5) Customer has contacted required MSS personnel to begin activation

As set forth below, Internet Security Systems will credit Customer's account if Internet Security Systems fails to meet this guarantee. Customer is solely responsible for providing Internet Security Systems accurate and current contact information for Customer's designated points of contact as well as valid IP network addressing info rmation. If Internet Security Systems fails to meet this guarantee the Customer's account shall be credited one (1) month Monthly Service Fee of the Customer's Small Office Managed Firewall Service. The Customer may obtain no more than one credit per contracted service.