Design of An Intelligent System For Booking of Trunk Calls In Pak Army Exchanges

By

Muhammad Aamir Ali

A DISSERTION

Submitted to

Faculty of National University of Science and Technology (NUST)

In partial fulfillment of the requirements

For the degree of

MS COMPUTER SOFTWARE ENGINEERING

Department of Computer Science

2000

Dedicated

To

Mujahideen-i-Islam

Abstract

Corp of Signals plays an important role in the establishment/management of secure and reliable communication network within Pak Army. With the recent break through by acquiring most sophisticated communication infra structure, there is a dire need to optimally/efficiently utilize the resources. Keeping in view this scenario, a system is proposed to meet such requirements, that accepts and checks the format of the string entered by the subscriber from telephone pad, checks the duplication of number booked, priority of number, and rank status of the subscriber. The systems saves the number booked in the database in the respective logical priority queues and communicate the booking number to the subscriber and the same information is displayed on the screen as well. The system is designed to scan all the logical priority queues automatically after every 5 seconds and the number awaiting will be flashed for the operator's attention to connect the call. Two separate modules are designed to handle the booking of number and scanning of the database for the awaiting calls, and are scheduled in such a way that each module will be running in parallel and independent to each other.

The proposed work is a research oriented and learning based in which a prototype system design has been developed to meet the above mentioned requirements. The proposed project provides subscriber's satisfaction and eliminates the chances of operator's level manipulation to the minimum, the call's record management and updating is dealt by the database automatically for the future references. No such study or work has been carried out previously neither the field of Telephony API has yet been explored in any of the previous studies. For the development of the system Visual C++ is used as

front-end tool, where as MS Access and Telephony API works in the background.

Acknowledgement

I am humbly thankful to "*Almighty Allah*" for his blessing and giving me the wisdom, knowledge and understanding, without which I would not have been able to complete the thesis.

I wish to express my deepest appreciation to my thesis supervisor, Dr Dil Muhammad Akbar Hussain for his supervision, dedication and commitment with this research. He ably supervised the research and spent hours to review the script of this thesis. He has always been very flexible and supportive of new ideas and efforts.

My very special thanks are extended to all faculty members of the department of computer sciences, specially Head of Department Mr Waseem Suleman and the kind members of my guidance committee namely Mr M.Akhtar Iqbal, Mr Fazal Ahmad and Mr Zarrar Javaid whose very useful suggestion and critical review assisted me in widening the perspective of this project.

I thank all my colleagues, especially to Mr. Tariq Ismail, Mr. Farooq Azam, Mr. Athar Mohsin, Mr Ibrar Hussain and Mr Hashim for their moral support and encouragement for completion of this project.

I am heartily indebted to my mother and mother in law for their continued prayers, encouragement and patience for the successful completion of this thesis. My special thanks to my wife for her patience, immeasurable faith and confidence in me which has always been a source of inspiration and encouragement, and my innocent kids who displayed tremendous amount of patience during my project.

My sincere gratitude to the National University of Science and Technology and Military College of Signals for the provision of the facilities for this research.

Contents

CHAPTER 1 Introduction *1*

1.1 Background **1**

1.2 Existing procedure **2**

1.3 Problem Statement **2**

Problem solving approach **3**

1.5 Objectives **4**

1.6 Advantages **5**

CHAPTER 2 Modems and Communications Tools *7*

2.1 Introduction **7**

2.2 Modems and the UniModem Drivers for Win95 and WinNT **7**

2.3 A Quick Review of How Modems Work **8**

2.4 The Universal Modem Drivers and TAPI Service Providers **8**

2.5 Basic Data Modems **9**

2.6 Data Modems with Voice **10**

2.7 Telephony Cards **11**

2.8 Summary **12**

CHAPTER 3 Telephony API *13*

3.1 Introduction **13**

3.2 Telephony API **13**

3.3 TAPI Concepts **14**

3.4 The Telephony API Model **15**

3.4.1 Line Devices 16

3.4.2 Phone Devices 16

3.5 Typical Configurations **17**

3.5.1 Phone-based 17

3.5.2 PC-based 17

3.6 Architectural Overview **18**

3.6.1 Existing 16-bit applications link to TAPI. 19

3.6.2 Existing 32-bit applications link to tapi32.dll. 19

3.7 TAPI Software Architecture **19**

3.8 Synchronous and Asynchronous Operations **21**

Chapter 4 Database *23*

4.1 Introduction **23**

4.2 DAO (Data Access Object) **23**

4.3 Services Provided by the Microsoft Jet Database Engine 24
4.3.1 Tables 25
4.3.2 Fields 25
4.3.3 Records 25
4.3.4 Indexes 26
4.3.5 DaoRecordset 26
4.3.6 Dynaset 26
4.3.7 Snapshot 27
4.4 Microsoft Jet database engine (MJDE) 27
4.5 Programming DAO in C++ 29
4.6 How MFC Encapsulates DAO 31
4.7 Mapping of DAO Objects to MFC Classes 31
4.8 How MFC Accesses the Database Engine 32
Chapter 5 Implementation Issues 34
5.1 Introduction 34
5.1.1 Class CMSProjSet 36
5.1.2 Class CMSProjView 38
5.1.3 Class CTelephoneLine 41
5.1.4 Class CTelephoneSound 44
5.1.5 Class CDLGAddNewNumber 47
5.1.6 Class CDLGDirectoryView 48
5.1.7 Class CDLGExistingProperties 49
5.1.8 Class CDLGLockUnLock 50
5.1.9 Class CDLGChangePassword 51
5.1.10 Class CQueryDlg 52
5.1.11 Class CDialerDlg 52
Chapter 6 Conclusion 55
6.1 Thesis summary 55
6.2 Advantages 58
6.3 Limitations 59
6.4 Future work 60
Bibliography 61

List of Figures

Page No

1.1 Concept of Project …….3

3.1 Grouping of TAPI services……...15

3.2 TAPI Architecture ……20

3.3 The TAPI software architecture ……...21

3.4 Processing of TAPI events……...23

4.1 Table design view in MS Access……..26

4.2 Hierarchy of DBEngine ……...29

4.3 Working of Jet Database Engine …….30

4.4 Proprietary of Jet Database Engine……...31

5.1 Classes and Functions of MSProj in Visual C++ editor……...37

5.2 Run time view of class MSProjView……...40

5.3 Run time view of class CDLGAddNewNumber…….49

5.4 Run time view of class CDLGDirectoryView……..50

5.5 Run time view of class CDLGExistingProperties……51

5.6 Run time view of class CDLGLockUnLock……52

5.7 Run time view of class CDLGChangePassword…….52

5.8 Run time view of class CqueryDlg……...53

5.9 Run time view of class CdialerDlg……...54

List of Tables

Page No

4.1 MFC Classes and Corresponding DAO Objects ……...32

4.2 How MFC Manages DAO Objects Not Mapped to Classes …….33

5.1 Member Functions of class CMSProjSet ……..38

5.2 Member Variables of class CMSProjSet ……..39

5.3 Member Functions of class CMSProjView ……..41

5.4 Member Variables of class CMSProjView ……...42

5.5 Member Functions of class CtelephoneLine …….43

5.6 Member Variables of class CtelephoneLine …….45

5.7 Member Functions of class CtelephoneSound ……..46

5.8 Member Variables of class CtelephoneSound ……..48

5.9 Member Functions of class CdialerDlg …….55

CHAPTER 1

1

21Introduction

2.11.1 Background:

The computer is replacing the traditional way of working in every walk of life and getting popularity as well as giving reliable and faster results. These advanced and reliable features gives an encouragement to the software designers to develop applications which provide efficient and and reliable working environments. Telephonic communication is the most popular means of communication. In Pakistan Army it is one of the major task of Corps of signals to provide the services relating to telephonic communication. Although with the induction of modern equipment the services are much faster and reliable, but still there is room for improvement. Pakistan is economically a poor nation and replacing any existing system abruptly would be a great economical burden. However, if the existing system could be used with slight modification and lesser cost, with additionally faster and reliable results. PASCOM exchanges are inducted in Pakistan Army few years' back. These exchanges have many advanced features in them. However, the trunk booking handling technique is quite traditional. The booking activity can be replaced in such a way that instead of asking the operator to book a number if the subscriber feed his required number through his telephone to the computer placed at the telephone exchange. By adopting this computerized booking method many ills of the system can be removed. These disadvantages are discussed in the following sections.

~~2.2~~1.2 Existing procedure

The working of exchanges are of a traditional type and pays an extra load on the working staff, there is one shift dedicated only to deal with call booking, the job is to entertain the subscriber for booking a call, the procedure for booking is quite lengthy specially when number of calls increases, procedurally he asked the subscriber about his local number, the number he wanted to book, his rank and name, he writes all this information on a chit (known to be a ticket) along with date and time of booking, then he has to check whether this telephone number is authorized to book a call, after that he hand over this ticket to the operator who places it in a queue , Now after connecting the call the operator has to sign the respective ticket and write down the time of connection and reason for not connecting in case call cannot be connected and then hand it over to the exchange supervisor, at the end of the day the supervisor has to maintain the record of all calls in a register for future references, it is a time consuming procedure . Moreover, if any complaint is raised then searching of a specified ticket from that junk is also very time consuming.

~~2.3~~1.3 Problem Statement

As mentioned traditional way of booking is lengthy / laborious and error prone, some of problems are highlighted as follows:

a. The subscriber has to remind the booking supervisor for his call may be number of times.

b. Subscriber cannot get his booking number all the time he wants to know about.

c. Manipulation of calls at operator level is possible.

d. Exchange supervisor has to monitor all calls keeping in view their priority.

e. Trunk supervisor has to maintain a log of all calls booked, at the end of the day.

Formatted: Bullets and Numbering

f. A subscriber may enjoy unauthorized privileges by using his personal contacts with the exchange staff.

g. Service regarding booking of trunk calls cannot be denied to any number.

h. Duplication in booking of calls are very much possible.

i. Every body whether authorized to use the service for trunk calls or not have a direct access to exchange.

2.41.4 Problem solving approach

The concept of the project is to design a system in which the subscriber dials the exchange number and he will then be connected to a computer, removing an intervention of any booking staff. More to say that the computer

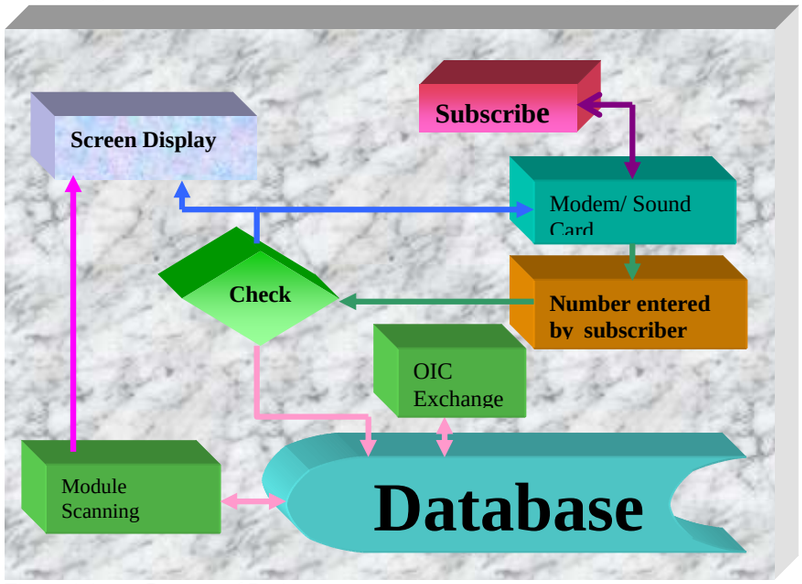


Figure 1.1 Concept of Project.

will play audible sound signal and communicate the booking number. The concept of the project is described as per the diagram below. The flow diagram gives a general view about the basic building blocks used in the project and the sequence of execution. Once the subscriber will dial the exchange number he gets connected with a PC. ~~Before login the caller~~After welcome message PC will first ask the user to enter his own number followed by the number he wants to book. After saving the number in database the booking number will be communicated to the subscriber where as another module keeps on scanning the database for awaiting calls in the queue. Only authorized caller will be allowed to enter the booking number. The communication between subscriber and PC is dealt through Telephony Application Programming Interface (TAPI), by utilizing Modem and sound card. Database is designed using Microsoft Access. The complete code of the project is written in Microsoft Visual C++.

~~2.5~~1.5 Objectives

To keep a constant check is impracticable and humanely impossible so there is a dire need to develop a transparent and user-satisfying environment, the objectives of the project are:

1. To develop an intelligent system of trunk calls booking, processing displaying, keeping in view the priority and rank status of the subscriber.
2. Trunk calls record keeping in database for future reference.
3. Reduction in the manpower employment.

~~2.6~~1.6 Advantages

Following advantages are fore seeable:

Subscriber's end:

1. It totally eliminates the interaction of subscriber with the exchange staff.
2. It gives a satisfaction to the subscriber that the number he booked will be serviced according to his priority.
3. Subscriber has the facility to get the information of his booking number at any time.
4. Subscriber has the facility to cancel his call if need be.
5. Almost eliminates the operator level manipulation regarding management of calls.

Management's end:

1. It totally eliminates the tedious job of exchange supervisor and call booking staff, which includes the management of call booking tickets as well as their record keeping.
2. Now the Operator's job is just dialing of flashed number. He is relieved of tickets handling, their sequencing and endorsement of service time, signatures and reason of not connecting the call in case any call could not be connected.
3. Duplication in call booking is not possible.

Chief Duty Signal Officer's (CDSO) end:

CDSO can enjoy the following privileges, just by the click of mouse:-

1. In case of any complaint, the CDSO can settle the issue by accessing the database within no time.
2. Can control the call's booking privilege from any number.
3. Can change the priority and rank status of in use numbers.
4. Can add new numbers in the database as and when required.
5. Can delete a number from the database if no more required.
6. All the settings are protected by a password for security purposes.

7. The manpower relieved by the developed system can be employed elsewhere.

CHAPTER 2

32Modems and Communications Tools

3.12.1 Introduction

This chapter discussed the differences between the three primary types of telephony hardware for PCs such as Basic data modems, Voice-data modems and Telephony cards. These three types of interface cards provide a wide range of telephony service for desktop workstations. The advantages and limitations of each of the card type and their use in telephony applications are discussed in this chapter.

Basic data modems can support Assisted Telephony services (outbound dialing) and are usually able to support only limited inbound call handling.

Voice-data modems are a new breed of low-cost modems that provide additional features that come close to that of the higher-priced telephony cards. These modems are usually capable of supporting the Basic Telephony services and some of the Supplementary services. The key to success with voice-data modems is getting a good service provider interface.

Finally, telephony cards offer the greatest level of service compatibility. Telephony cards usually support all of the Basic Telephony and all of the Supplemental Telephony services, including phone device control. Most telephony cards also offer multiple lines on a single card. This makes them ideal for supporting commercial-grade telephony applications.

3.22.2 Modems and the UniModem Drivers for Win95 and WinNT

All TAPI services are routed through some type of modem. These modems also depend on the Windows operating system to supply device drivers to communicate between programs and the device itself. While a detailed discussion of device drivers is beyond the scope of this chapter it is a good idea to have a general understanding of how Windows uses device drivers and how modems work. In this section we'll take a quick review of modem

theory and a short discussion of the Universal Modem Driver that ships with Win95 and WinNT.

~~3.3~~2.3 A Quick Review of How Modems Work

Before getting into the details of how the three types of telephony hardware differ, it is important to do a quick review of how modems work.

Sending computer data over voice-grade phone lines is a bit of a trick. All data stored on a PC (documents, programs, graphics, sound and video files, and so on) is stored as binary data. However, standard telephone lines are not capable of sending binary data. Which means that any information sent over the telephone line has to be in the form of sound waves. In order to accomplish this feat, hardware is used to convert digital information into sound (that is, to modulate it), then back again from sound into digital information (demodulate it). This process of modulating and demodulating has lead to its name: mo-dem (modulate-demodulate).

Sending data over phone lines involves three main steps. First, a connection must be established between two modem devices over a telephone line. This is typically done by having one modem calls the other modem. If the second modem answers the telephone call, the two modems go through a process of determining if they understand each other called handshaking. If that is successful, then information can be passed.

In the second step, the digital information is modulated into sound and then sent over the voice-grade telephone line to the second modem. In the last step, the modem at the other end of the call converts (demodulates) the sound back into digital information and presents it to the computer for processing (view the graphic, save the file, play the video or audio, and so on).

~~3.4~~2.4 The Universal Modem Drivers and TAPI Service Providers

TAPI requires each workstation to have not just a TAPI-compliant application, but also a Telephony Service Provider Interface (TSPI). This

interface talks directly to the hardware to convert TAPI service requests into commands understood by the hardware. The hardware vendor usually supplies the TSPI, but Microsoft Win95 ships with a simple TSPI called the UniModem Driver (Universal Modem Driver). The UniModem driver is designed to support Assisted Telephony and some Basic Telephony. Programmer can build simple applications that allow users to place and receive voice and data calls using basic data modems and the UniModem driver that ships with Win95 and WinNT.

Microsoft has released a modem driver that supports additional voice features including playing and recording audio files. This driver is called the UniModemV Driver (Universal Modem for Voice). This driver supports the use of voice commands along with recording and playing back voice files. It can also handle caller ID and some other added service features. Exactly what the UniModemV driver can do is also dependent on the hardware. The telephony hardware must recognize any advanced features and be able to communicate them to the driver.

~~3.5~~2.5 Basic Data Modems

The most basic type of hardware that supports TAPI is the basic data modem. This type of modem is designed to use analog phone lines to send digital data. Any computer that can access online services (BBS, Internet, commercial information services, and so on) has at least this level of modem hardware. Programmer can get basic data modems with speeds of 9600 to 14,400bps (bits per second) for $100 U.S. or less. Almost all basic data modems recognize a common set of control codes. This set of control codes is called the Hayes or AT command set. The makers of the Hayes modem has developed this set of controls. The first command in the set (AT) is the "attention" command. This tells the device that the application is about to send control codes directly to the hardware. The command set is known by the author's original name ("Hayes") or by the first command in the set ("AT").

Basic data modems support Assisted Telephony services without any problem (that is, placing outbound calls). Most basic modems are capable of supporting some of the Basic Telephony services, including accepting inbound calls. However, if programmer wants to perform any of the more advanced TAPI services, such as playing or recording audio files, need more advanced hardware. Also, if programmer wants to access advanced features available for voice telephones such as caller ID, call hold, park, forward, and so on, in such a case the requirement is more than a basic data modem.

~~3.6~~2.6 Data Modems with Voice

There is a new type of modem available that offers all the services of a data modem, but also has added support for voice services. These modems are often called voice-data modems (or data-voice modems). This hardware has additional program built into the chips that can support advanced telephone features such as caller ID, call hold, park, forward, and so on. Just as basic data modems use the AT command set, the voice-data modems use an extension of that set called the AT+V command set (AT plus Voice).

Voice-data modems also require a TAPI-compliant modem driver in order to work with TAPI services. This driver is usually supplied by the hardware vendor. Microsoft also supplies a modem driver that supports voice services-the UniModemV driver. If any modem does not ship with a TAPI-compliant driver, one has to install the UniModemV driver to enable voice features.

A word of advice when purchasing a voice-data modem. There are several modems in the market that offer voice, voice-mail, telephone answering, and other TAPI-like services for PCs but not all of them are TAPI-compliant. Which essentially means that programmer may be able to do many things to his liking but he may not be able to program it using TAPI services.

There are a handful of voice-data modem vendors that have announced the release of TAPI-compliant hardware. Here is a list of some vendors currently offering TAPI-compliant voice-data modems:

1. Compaq Presario Systems
2. Creative Labs Phone Blaster
3. Logicode 14.4 PCMCIA
4. Diamond Telecommander 2500
5. Cirrus Logic
6. Aztech Systems

Voice-data modems with supporting TAPI drivers offer a wide range of access to TAPI services. Programmer can use voice-data modems to perform both outbound and inbound call handling, play and record voice files, and (if the feature is available on the telephone line) support caller ID and other advanced services for single-line phones.

~~3.7~~2.7 Telephony Cards

The most advanced level of hardware a programmer can get for TAPI services on a desktop PC is a dedicated telephony card. This is a piece of hardware especially made to handle telephone services. Most telephony cards are designed to handle more than one line at a time. If programmer is planning an application that must answer several phone lines or perform any line transfers, and so on, telephony card is the solution.

Most telephony cards are sold as part of a kit. It can get software development tools, cards for the PC, cables, and documentation all for one price. This price usually starts at around $1000 U.S. [2.1] and can easily climb depending on the number of lines programmer wish to support. Even though the price is a bit high but its advantages are enormous.

As with other telephony hardware, telephony cards need an accompanying TAPI driver in order to recognize TAPI calls from user's application. While most telephony card vendors are working on TAPI drivers, not all of them supply one. It is important to check the specifications of the hardware and supporting materials before buying.

Telephony cards (along with TAPI drivers to match) offer the greatest access to TAPI services. Programmer can support all the Assisted TAPI and

Basic TAPI functions along with access to Supplemental TAPI services. Also, if the driver supports, it may be able to use Extended TAPI services to gain access to vendor-specific functions unique to the installed hardware.

~~3.8~~2.8 Summary

TAPI is a new subject and it has an important role in the implementation of this project. Secondly this subject has not yet touched during any of the previous courses. Keeping in view its applications and development requirements of telephony based projects, it is discussed in detail, to benefit in future research. In this chapter programmer learned the differences between three types of hardware options and how they stand in offering support for TAPI services on PC workstations. Basic data modems support Assisted Telephony services (outbound dialing) and can support only limited inbound call handling. Use of this type of hardware is recommended for developing simple outbound dialing applications. Voice-data modems are capable of supporting the Assisted Telephony and Basic Telephony services and many of the Supplementary services. Use of this type of hardware is recommended to provide both inbound and outbound services on a single-line phone. Telephony cards support all of the Basic Telephony and all of the Supplemental Telephony services, including phone device control. Most telephony cards also offer multiple lines on a single card. This makes them ideal for supporting commercial-grade telephony applications. This chapter got a quick review of modems and modem drivers and that Win95 and WinNT rely on the UniModem or UniModemV modem drivers to communicate between the telephony hardware and user's applications program. It also highlights that, no matter what hardware programmer purchase, programmer will need a TAPI-compliant TSPI (Telephony Service Provider Interface) that matches the hardware purchased. Hardware vendors may recognize the UniModem or UniModemV drivers, or ship their own TSPI drivers with their hardware.

CHAPTER 3

43Telephony API

Formatted: Bullets and Numbering

4.13.1 Introduction

The *Telephony Application Programming Interface* (TAPI) is one of the most significant API sets to be released by Microsoft. The telephony API is a single set of function calls that allows user's to manage and manipulate any type of communications link between the PC and the telephone line(s). While telephony models for the PC have been around for several years, the telephony API establishes a uniform set of calls that can be applied to any type of hardware that supplies a TAPI-compliant service provider interface (SPI).

Telephony Application Programming Interface (TAPI) provides a consistent programming interface for a variety of devices operating on voice grade lines. The devices include modems, FAX modems, voice capable modems, computer-controlled telephone sets, and many more. TAPI provides services for placing outgoing calls, accepting incoming calls, and managing calls and devices. This chapter describes a general overview of the Telephony API and how it fits into the WOSA (Windows Open Services Architecture) model.

4.23.2 Telephony API

The Telephony Application Programming Interface (TAPI) is an application-programming interface that is used to communicate by means of telephones. API is a set of routines that an application program uses to request and carry out lower-level services performed by a computer's operating system. It is a single set of function calls that allows programmers to manage and manipulate any type of communications link between the PC and the telephone line. Telephony services are divided into Assisted Telephony services and the services provided by the full Telephony API. In general, the full Telephony API is used to implement powerful telephonic applications and Assisted

Telephony is used to add minimal but useful telephonic functionality to non-telephony applications. Telephony services are divided into the categories shown in the following illustration:

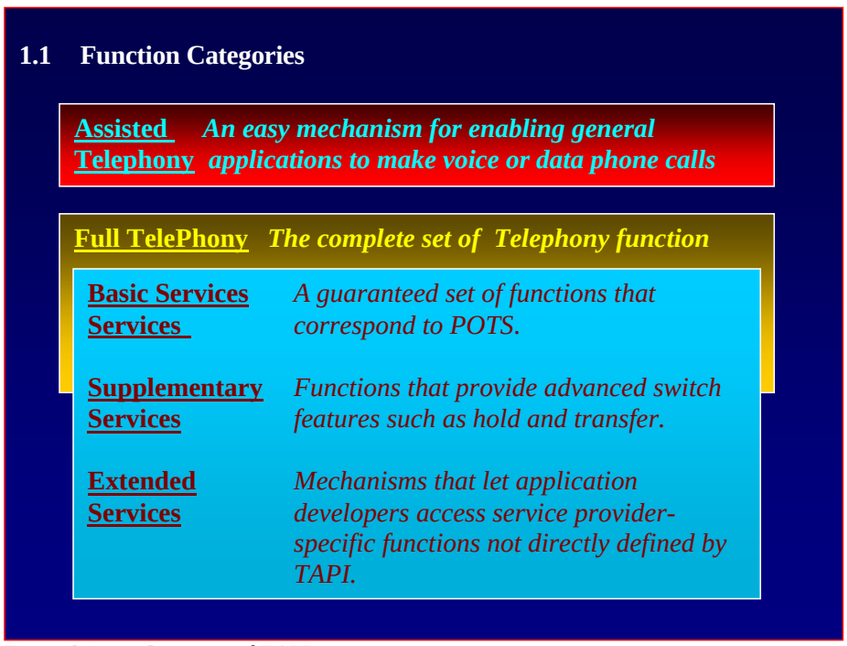


Figure 3.1 Grouping of TAPI services.

~~4.3~~3.3 TAPI Concepts

TAPI provides a series of personal telephony services. Telephony, in this context, refers to technology in general that connects computers with the telephone network.

TAPI services provide all aspects of usage of the telephone network. This includes connecting to the network, placing and accepting calls, call

management features (such as transferring calls, setting up conference calls), use of calling number identification (Caller ID) for identifying incoming calls, and more.

TAPI services are divided into basic, supplementary, and extended services. Basic services are generally supported by all devices, supplementary services may only be available on special devices. Extended services are provider-specific.

For example, TAPI can place a call on all telephone lines; however, call management functions, such as transferring a call, may only be available on devices that specifically support such a feature, and thus is considered a supplementary service.

TAPI is not restricted to what is whimsically referred to by the acronym POTS (Plain Old Telephone Service). POTS is analog service on the *local loop* (the wire connecting the telephone set with the nearest switching office). POTS supports voice calls with a 3.1 kHz bandwidth, or data calls at speeds up to 28.8 kbps using V.34 modems.

In contrast, ISDN (Integrated Services Digital Network) supports up to 128 kbps with its *Basic Rate Interface* (BRI-ISDN); the speed on PRI-ISDN (Primary Rate Interface) is much higher. TAPI supports ISDN as well as other connection types, such as switched 56, or T1/E1. TAPI can also utilize CENTREX features and the features of Private Branch Exchanges (PBXs).

~~4.4~~3.4 The Telephony API Model

The telephony API model is designed to provide an abstracted layer for access to telephone services on all Windows platforms. In other words, the telephony API is a single set of functions that can be used to access all aspects of telephony services within Windows operating system. The aim of TAPI is to allow programmers to write applications. Applications written using TAPI to gain direct access to telephone-line services work in the same way on analog or digital phone lines. Applications that use TAPI can generate a full set of dialing tones and flash-hook functions. The TAPI design model is divided into two

areas, each with its own set of API calls. The two TAPI devices are, *Line devices*, which models the physical telephony lines used to send and receive voice and data between locations, and *Phone devices* to model the desktop handset used to place and receive calls.

3.4.1 Line Devices

The line device is used to model the physical telephone line. It is important to understand that, in TAPI, the line device is not really a physical line; it is just a model or object representing a physical line. In TAPI applications, a program can keep track of several line devices, each of which is connected to a physical line. The same TAPI application could also keep track of multiple line devices their number is more than the total physical lines available to the PC. For example, a single TAPI application could be designed to provide voice, fax, and data links for a user. The TAPI application would identify three line devices. One for voice calls, one for fax transmission, and one for sending and receiving data via an attached modem. If the PC has only one physical phone line attached, the TAPI application would share the one line between the three defined line devices. This is called dynamic line mapping. Each time the TAPI application starts a line device, it requests the first available physical line that has the capabilities needed (voice, fax, data, and so on). If a line is not available, a message to that effect is returned to the calling program. In some cases, such as fax transmissions, the TAPI application may "queue up" the line request for processing at a later time. If two lines are available, the TAPI application utilize them as they are needed. If a third line device becomes active, the TAPI application knows that there are no other available open lines and notifies the user (or possibly queues up the outbound call for later).

3.4.2 Phone Devices

The second type of device modeled by TAPI is the phone device. This model allows TAPI programmers to easily create "virtual phones" within the PC workspace. For example, a standard PC with a sound card, speakers, and

microphone can emulate all the functions of a desktop phone. These virtual phones, like their line device counterparts, need not exist in a one-to-one relationship to physical phones. A single PC could model several phone devices, each with its own unique characteristics. When an actual call is made, the user could select one of the phone devices, enter the desired number and then the TAPI application would attach the phone device to an available line device.

~~4.5~~3.5 Typical Configurations

The TAPI model is designed to function in several different physical configurations, each of these schemes have advantages and drawbacks. There are four general physical configurations:

~~4.5.1~~3.5.1 Phone-based

This configuration is best for voice-oriented call processing where the standard handset (or some variation) is used most frequently. In phone-based TAPI configurations, the standard telephone handset is connected to the telephone switch and the PC is connected to the telephone. This configuration is most useful when the telephone handset is the primary device for accessing the telephone line. Since the telephone rests between the PC and the switch, the PC may not be able to share all the activities on the line. A phone-based configuration does not preclude the use of the PC to originate calls. As long as the PC is equipped with a phone card that allows dialing, the PC can originate a call and then allow the handset to pick up that call at any time.

~~4.5.2~~3.5.2 PC-based

PC-based configuration is best for data-oriented call processing where PC is used most frequently for either voice or data processing. PC-based TAPI configurations place the PC between the telephone switch and the standard handset. This configuration is most useful when the PC is primary device for accessing the telephone line. In this configuration, mostly PC originates phone calls. Typically, this is done via a phone card and software on the PC that

manages a list of phone numbers and handles the dialing of the phone. Depending on the exact media mode of the call, PC can be used to display digital data on screen while handling voice information at the same time. Users could originate a voice call through the handset and then switch to the PC to capture and display digital data sent over the same line. Another major advantage of the PC-based configuration is that the PC can act as a call manager for the handset. This is especially valuable in a mixed-mode environment where voice, data, and fax are all coming in to the same phone address. For example, when a call comes in to the attached phone line, the PC can answer the call and determine the media mode of the call. If it is a fax call, the PC can route the call directly to an attached fax machine (or to the fax driver on the PC). Data calls can be handled directly by the PC and voice calls can be forwarded to the attached handset.

~~4.6~~3.6 Architectural Overview

Win32 Telephony is a full, 32-bit implementation of the original 16-bit Windows Telephony application and service provider interface. Other than components provided for backward compatibility, all components of Win32 Telephony, including service providers, are implemented in 32 bits. The following figure illustrates the architecture of Win32 Telephony on the Windows operating systems.

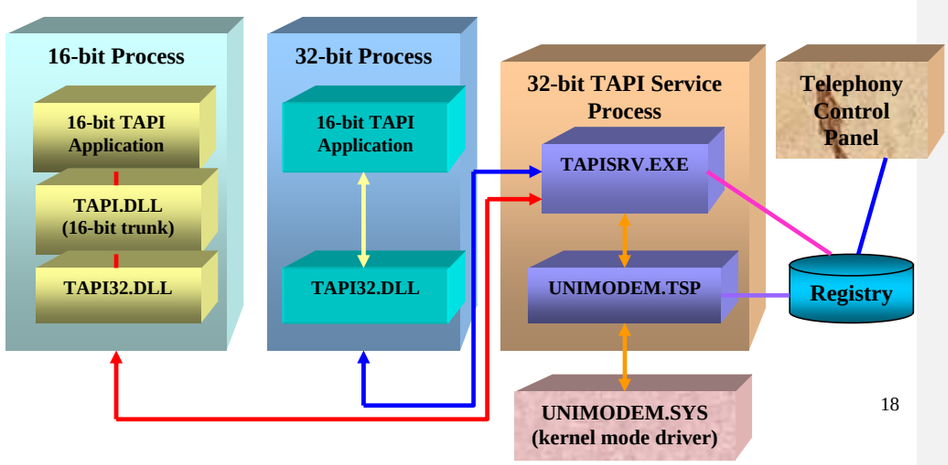


Figure 3.2 TAPI Architecture

~~4.6.1~~3.6.1 Existing 16-bit applications link to TAPI.

In Windows 3.1 and Windows 95, TAPI is the core of Windows Telephony. Under Windows NT/Windows 2000, TAPI is simply a thunk layer to map 16-bit addresses to 32-bit addresses, and pass requests along to tapi32.dll. (Thunk is a small section of code that performs a translation or conversion during a call or indirection. For example, a thunk is used to change the size or type of function parameters when calling between 16-bit and 32-bit code.)

~~4.6.2~~3.6.2 Existing 32-bit applications link to tapi32.dll.

In Windows 95, tapi32.dll is a thunk layer to TAPI. In Windows 98 and Windows NT/Windows 2000, tapi32.dll is a thin marshaling layer that transfers function requests to tapisrv.exe and, when needed, loads and invokes service provider user interface DLLs in the application's process, (Marshaling is defines as the packaging and sending interface method calls across thread or process boundaries).

~~4.7~~3.7 TAPI Software Architecture

The heart of TAPI is the TAPI dynamic link library (DLL) that offers TAPI services to the applications. This DLL serves as a layer between telephony applications and TAPI service providers. One such service provider is the UNIMODEM driver; this Universal Modem driver is supplied with Windows 95 and provides TAPI services for modems compatible with the Hayes AT command set . This basic TAPI architecture is shown in Fig 3.2.

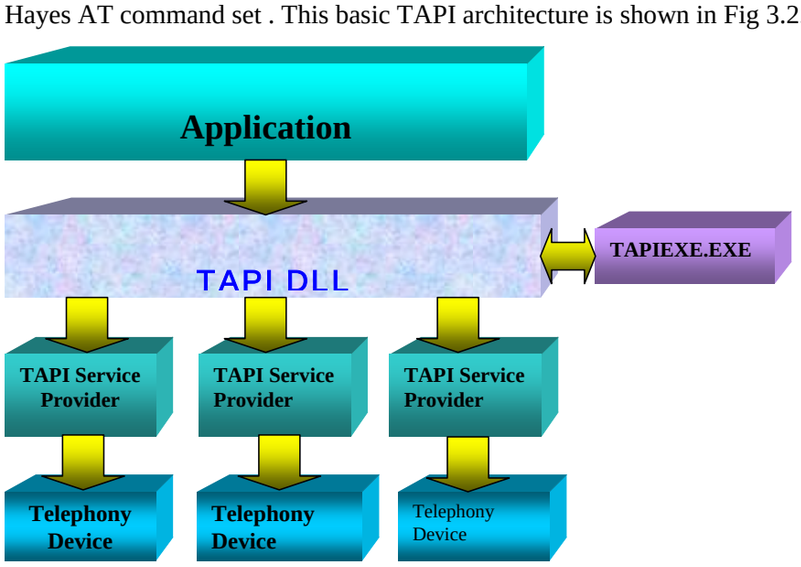


Figure3.3 The TAPI software architecture.

In addition to the TAPI DLL and the telephony service providers (drivers), another important, component of TAPI is the executable program tapiexe.exe. This program plays an important role when TAPI sends notifications to the calling application via callback functions.

~~4.8~~3.8 Synchronous and Asynchronous Operations

Many TAPI operations are synchronous; that is, when the TAPI function returns, the operation is either completed or failed, in this case an error code is returned. However, some TAPI operations are asynchronous; the TAPI function returns indicating whether the TAPI operation has been successfully initiated, but the operation is completed in another thread, and the application is notified via a callback function. The callback function is registered with TAPI when the TAPI library is initialized. The actual callback mechanism deserves a closer examination, especially because it has some consequences as to how TAPI functions operate. When a service provider wishes to place a notification, it calls the TAPI DLL. In effect, it requests that the DLL notify all concerned applications that a specific event has taken place. This first call to the TAPI DLL takes place in the execution context of the service provider. The TAPI DLL in turn sends a message to tapiexe.exe. This executable program calls the TAPI DLL itself, this time in its own execution context. This call instructs the TAPI DLL to post a Windows message to the applications needed to be modified. When the application receives and processes the message in its message loop, the message is dispatched to the TAPI DLL again, this time in the application's execution context. The TAPI DLL may in turn call the application's registered TAPI callback function to notify the application of a TAPI event. The TAPI notification mechanism is shown in Fig 3.4.

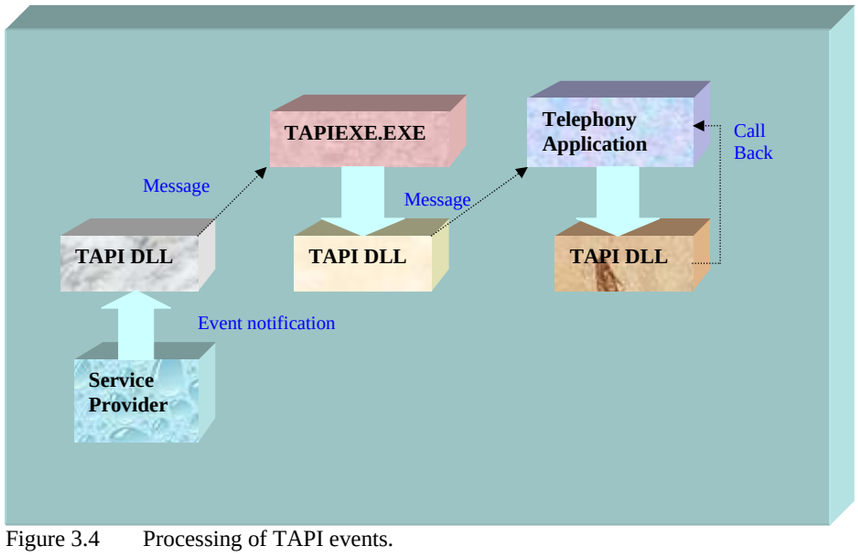


Figure 3.4 Processing of TAPI events.

This mechanism has important implications for the architecture of TAPI applications. The scenario described here makes it clear that TAPI applications must have a message loop in order to process notifications correctly. Although the use of a callback function may imply that a message loop is unnecessary, this is not the case; the callback function is only called after the application receives a Windows message that the TAPI DLL processes. Another consequence concerns the use of multiple threads. It is important to realize that in order for TAPI to

operate as expected threads that call asynchronous TAPI functions must have a message loop. The callback function is called in the context of the thread making the asynchronous call; this cannot happen unless the thread processes Windows messages.

Chapter 4

~~5~~4Database

~~5.1~~4.1 Introduction

Data Access Objects, or DAOs, is Microsoft's latest invention in database access technology. This technology with the help of a set of specialized MFC classes is used for database access in Microsoft's Visual C++. Data Access Objects enable the programmers to access and manipulate databases through the Microsoft Jet database engine. Through this engine, programmers can access data in Microsoft Access database files (MDB files).

~~5.2~~4.2 DAO (Data Access Object)

Data Access Objects enable the programmers to access and manipulate database through the Microsoft Jet database engine. Through this engine, programmers can access data in Microsoft Access database files (MDB files). The technology also enables the programmers to access local and remote databases through ODBC (Open Database Connectivity) drivers. Data Access Object technology is based on OLE (Object Linking and Embading). Figure 4.1 depicts the hierarchy of Data Access Objects. This hierarchy is greatly simplified by the DAO classes in MFC. Many DAO functions utilize Structured Query Language (SQL) statements. One can use the SQL SELECT statement to retrieve data from a database, or the SQL UPDATE, INSERT, and DELETE statements to modify the contents of the database. The easiest way to create SQL statements to use with DAO objects is to create the query from within Microsoft Access, save the query in the database, and access the query through a QueryDef object. Visual C++ provides extensive support for building DAO

Figure 4.3 DAO Object Hierarchy. applications through the AppWizard. In addition to ODBC, AppWizard enables the programmers to create applications that are based on DAO Classes.

Both Data Access Objects (DAO) and Open Database Connectivity (ODBC) are application programming interfaces (APIs) that provides the ability to write applications that are independent of any particular database management system (DBMS). DAO is familiar to database programmers using Microsoft Access or Visual C++. DAO uses the Microsoft Jet database engine to provide a set of data access objects, database objects, tabledef, querydef objects, recordset objects, and others. DAO works best with xxx.MDB files like those created by Microsoft Access, but the programmers can also access ODBC data sources through DAO and the Microsoft Jet database engine.

5.34.3 Services Provided by the Microsoft Jet Database Engine

The Microsoft Jet database engine can be thought of as a data manager component with which other data access systems, such as Microsoft Access, Microsoft Visual Basic and Visual C++ are built. Microsoft Jet provides a variety of database management services, including data definition, data manipulation, data integrity, and security. There are seven basic services that a DBMS should provide. The seven basic functions of a DBMS are:

a. Data definition and integrity. Create and modify structures for holding data, such as tables and fields, and ensure that rules to prevent data corruption from invalid entries or operations are applied to data operations.
b. Data storage. Store data in the file system.
c. Data manipulation. Add new data, modify or delete existing data.
d. Data retrieval. Retrieve data from the system.
e. Security. Control users' access to database objects and data, thereby protecting the objects and data against unauthorized use.
f. Data sharing. Share data in a multiuser environment.
g. Database maintenance. Compacts, repair, and convert data and database objects.

~~5.3.1~~4.3.1 Tables

The Project database has various tables to group information; for example,View, Directory, Password. Tables are made up of rows and columns of related information. In most databases, rows are referred to as “records”, and columns are referred to as “fields”.

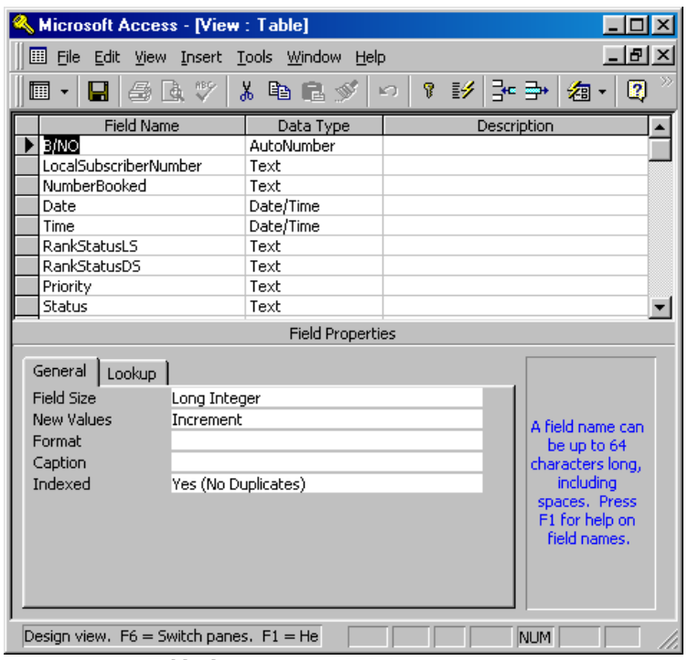


Figure 4.1 Table design view in MS Access.

~~5.3.2~~4.3.2 Fields

Each field in a table contains a single piece of information. For example the table View looks like shown in the figure 4.2.

~~5.3.3~~4.3.3 Records

A record contains information about a single entry in a table.

~~5.3.4~~4.3.4 Indexes

An index helps the application to find specific records and to sort records faster. Typically, one can index fields in a table that are used repeatedly to search for data. However, indexes slow down the editing, adding, and deleting of data so they should be used sparingly.

~~5.3.5~~4.3.5 DaoRecordset

A DaoRecordset, is an object that contains a set of records from the database. Some recordset types are based on single tables. More commonly, however, recordsets are based on a query that is created by using Structured Query Language (SQL). A table-type recordset is based on a single table where as SQL-based query can be used to retrieve a specific subset of records, or to retrieve information from two or more tables at once. SQL enables the user to specify which records to retrieve and the order in which to retrieve them. There are three types of Recordset objects. The type of recordset that is created should be based upon what a user intend to do with the data.

~~5.3.6~~4.3.6 Dynaset

Dynaset-type recordsets provide more flexibility than tables. A dynaset can refer to any table, attached table, or query. A dynaset provides an updateable view to the data. As the application scrolls to a changed record, a new copy is retrieved, bringing it up to date. This dynamic behavior is ideal for situations in which it is important to be completely up to date. A dynaset contains a set of keys to the underlying tables. The actual data is retrieved when the user accesses a particular record. Because a keyset is used to refer data, a dynaset reflects modifications to existing records modified by other users, but does not reflect new records added by other users. Similarly, if another user deletes a record after the application's owner dynaset is fully populated, then dynaset will contain a pointer to the deleted record and the application's owner receives a runtime error if he attempts to access a deleted record. A dynaset is not fully populated until move to the end of the recordset.

~~5.3.7~~4.3.7 Snapshot

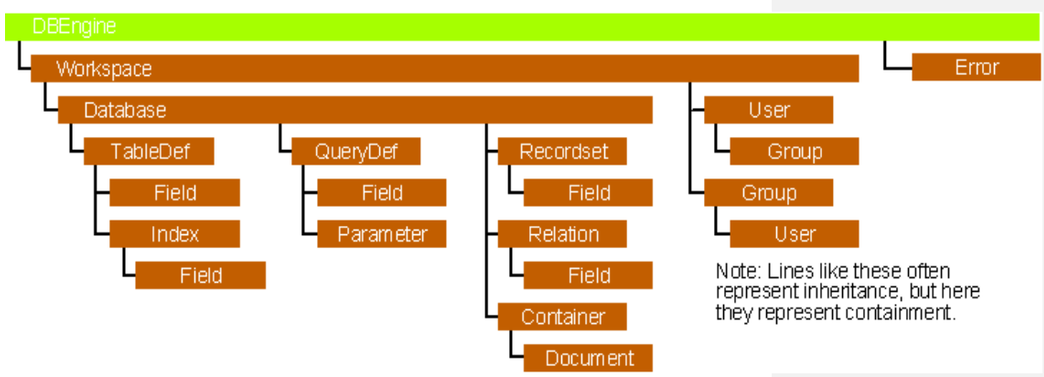
A "snapshot" reflects the state of the data at a particular moment, the moment the snapshot is taken. This behavior is ideal for reporting. As it takes time to retrieve the records for a snapshot, the moment at which the snapshot occurs is not instantaneous.

~~5.4~~4.4 Microsoft Jet database engine (MJDE)

The part of the Microsoft Access database system that retrieves data from and stores data in user and system database. The Microsoft Jet database engine can be thought of as a data manager upon which database systems, such as Microsoft Access are built.

The Jet 3.0 Engine, which is used in Access, Visual C++, and other Microsoft applications, is an efficient tool for storing and accessing data. It uses a cost-based query optimizer that takes programmer's query and SQL statements and finds what it thinks is the best plan for executing the query. It also includes many other features, such as a cache and a read-ahead cache, which it uses to store data

DAO (Data Access Objects) is a set of OLE objects that simplifies database programming. There's a Database object to represent what else the database, which contains a collection of Tabledef (table definition) objects, each of which in turn contains a collection of field objects. Each object has properties and methods that expose pertinent functionality. Figure 4.3 shows the object hierarchy.



a.

Figure 4.2 Hierarchy of DBEngine.

DAO uses a set of DLLs called the Jet engine. These DLLs provide access not only to MDB files, but also to other database formats including Xbase formats such as dBase and FoxPro, the Paradox DB format, spreadsheet data from Microsoft Excel and Lotus 1-2-3, and common text file formats such as fixed-width and comma-separated text. Jet engine even provides a route to ODBC sources.

Jet includes a powerful query processor that makes complex operations easy. For example, programmer can query an MDB table joined to a SQL Server able and update the resulting view. The ability to create stored queries (equivalent to a SQL view) that in turn reference other queries, all of which are still updatable, is quite powerful. It removes much of the application development burden.

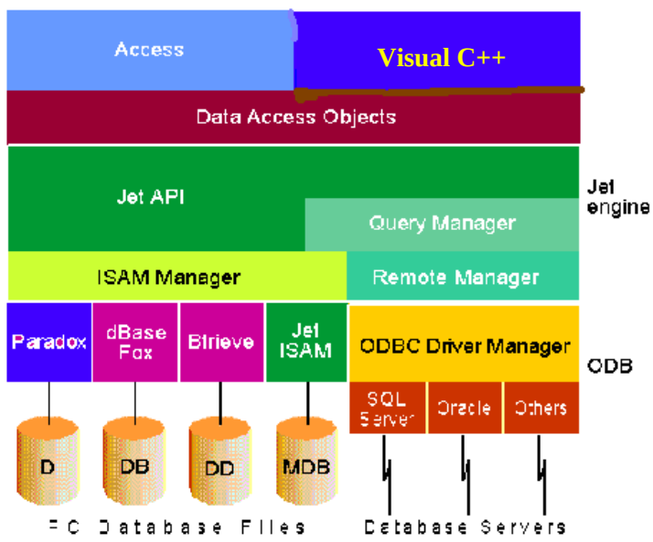


Figure 4.3 Working of Jet Database Engine

~~5.5~~4.5 Programming DAO in C++

There are three ways programmer can program DAO from C++:

1. Invoke the OLE automation dual interfaces directly.
2. Use the MFC CDao classes.
3. Use the DAO SDK classes.

Naturally, each of these methods has advantages and disadvantages. Since programming directly through automation is fairly tedious, most people opt for either the MFC or DAO SDK classes. The MFC classes (CDaoXxx) are designed to have the same look and feel as the MFC ODBC classes (CDatabase), with some notable improvements such as the ability to specify field names at run time.

Which classes are used is up to the programmer. Programmer can even mix classes within a single app. It might use the MFC classes for routine forms and

viewing, saving the DAO SDK classes for more specialized features like user-defined properties. Except where DAO is required for advanced features, the choice is more a matter of personal taste and experience than any technical merit, although in general the DAO SDK classes are more efficient since they map directly to the underlying objects and methods; whereas a single MFC call might translate into many DAO interface calls.

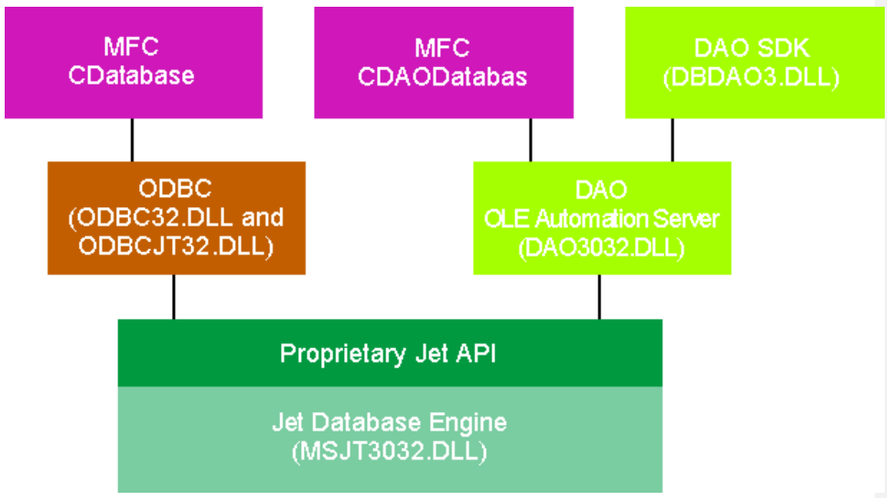


Figure 4.4 Proprietary of Jet Database Engine.

The Microsoft Foundation Class Library is a C++ programming-language framework that is primarily designed to create Microsoft Windows-based applications. This collection of C++ classes goes beyond a standard C++ library by providing some of the most basic structural elements to create programs. This framework facilitates the handling of data and much of the user interface that is characteristic of this type of program. This expert point of view outlines the reasons for learning and using the Microsoft Foundation Class library.

~~5.6~~4.6 How MFC Encapsulates DAO

The MFC DAO classes treat DAO much as the MFC classes for programming Windows treat the Windows API. MFC encapsulates or "wraps," DAO functionality in a number of classes that correspond closely to DAO objects. Class CDaoWorkspace encapsulates the DAO workspace object, class CDaoRecordset encapsulates the DAO recordset object, class CDaoDatabase encapsulates the DAO database object, and so on.

MFC's encapsulation of DAO is thorough, but it is not completely one-for-one. Most major DAO objects do correspond to an MFC class and the classes supply generally thorough access to the underlying DAO object's properties and methods.

~~5.7~~4.7 Mapping of DAO Objects to MFC Classes

The tables 4.1 and 4.2 show as to how DAO objects correspond to MFC classes.

Table 4.1 MFC Classes and Corresponding DAO Objects.

Class	*DAO object*	*Remarks*
CdaoWorkspace	Workspace	Manages a transaction space and provides access to the database engine.
CdaoDatabase	Database	Represents a connection to a database.
CdaoTableDef	Tabledef	Used to examine and manipulate the structure of a table.
CdaoQueryDef	Querydef	Used to store queries in a database. Programmer can create recordsets from a querydef or use it to execute action or SQL pass-through queries.
CdaoRecordset	Recordset	Used to manage a result set, a set of records based on a table or selected by a query.
CdaoException	Error	MFC responds to all DAO errors by throwing exceptions of this type.

Class	DAO object	Remarks
CdaoFieldExchange	None	Manages exchange of data between a record in the database and the field data members of a recordset.

Table 4.2 How MFC Manages DAO Objects Not Mapped to Classes.

DAO object	How MFC manages it
Field	Objects of classes CdaoTableDef and CDaoRecordset encapsulate fields and supply member functions for adding them, deleting them, and examining them.
Index	Objects of classes CdaoTableDef and CdaoRecordset encapsulate indexes and supply member functions for managing them. Tabledefs can add, delete, and examine indexes. Tabledefs and recordsets can set or get the currently active index.
Parameter	Objects of class CdaoQueryDef encapsulate parameters and supply member functions for adding them, deleting them, examining them, and getting and setting their values.
Relation	Objects of class CdaoDatabase encapsulate relations and supply member functions for adding, deleting, and examining.

~~5.8~~4.8 How MFC Accesses the Database Engine

DAO has a DBEngine object that represents the Microsoft Jet database engine. The DBEngine object provides properties and methods which programmer can use to configure the database engine.In MFC, there is no

DBEngine object. Access to important properties of the database engine is supplied via class CDaoWorkspace. To set or get these properties, call any of the static member function of CDaoWorkspace.

Chapter 5

65Implementation Issues

6.15.1 Introduction

This project has two basic components, *one which interact with the user* and *other with the database*. These components are namely TAPI and CDaoRecordset. Database *keeps all the information about user as his telephone number, rank, priority etc* where as TAPI *will communicate with the user through modem.* All necessary information relating to each number is also being maintained by the database. Figure 1.1 already explained the conceptual working of this project in the form of block diagram.

A separate module has been developed which works in parallel with the above mentioned module. It intelligently process the number booked according to their priority and rank status and prompts the operator to dial accordingly.

Modules communicate with the *database* through Data Access Object (DAO) classes of MFC tool. DAO classes provide more power and flexibility to handle databases through MJDE (Microsoft Jet Database Engine).

Once the user dial the number the computer answers after three bells and ask the caller to enter his required number, the user has to enter the string like:

5224*8221001234567*#	For booking
5224*8221001234567*2#	For Cancellation
“ 5224 ”	Own Number(As the project is designed for PASCOM exchanges which donot support CLI function that’s why own Number is required).
“ 8221 ”	City Code.
“ 00 ”	Army to civilian Code (only required for civilian numbers).
“ 1234567 ”	The required telephone number.

After getting this string it pass it through four levels as :

1. First Level will check the format of the string.
2. Second Level will see whether it is for booking or for cancellation.
3. Third level will check for duplication.
4. Forth Level will check the authorization of user.

After passing through these levels number will be booked and user will be communicated his booking number and the same information is displayed on the screen.

The implementation detail of project will be discussed under following section.

This project is a Dao-based application, developed through Application Wizard of Visual C++ 6. An extensive use of Microsoft Foundation Classes ((MFC) has been made because of its user friendly interface for the users. MFC programming is done through object oriented approach in which, different modules are encapsulated in different classes performing a specific task. Normally each dialog box and its visible objects are encapsulated in a separate class. Each class may have a number of member functions and member variables to execute the logic.

Project is developed with the combination of 19 classes and 127 user defined functions in all the classes shown in figure 5.1. This section will discuss only those classes and functions which are heart of the design. Each class will be discussed in a separate table along with its important member functions and member variables.

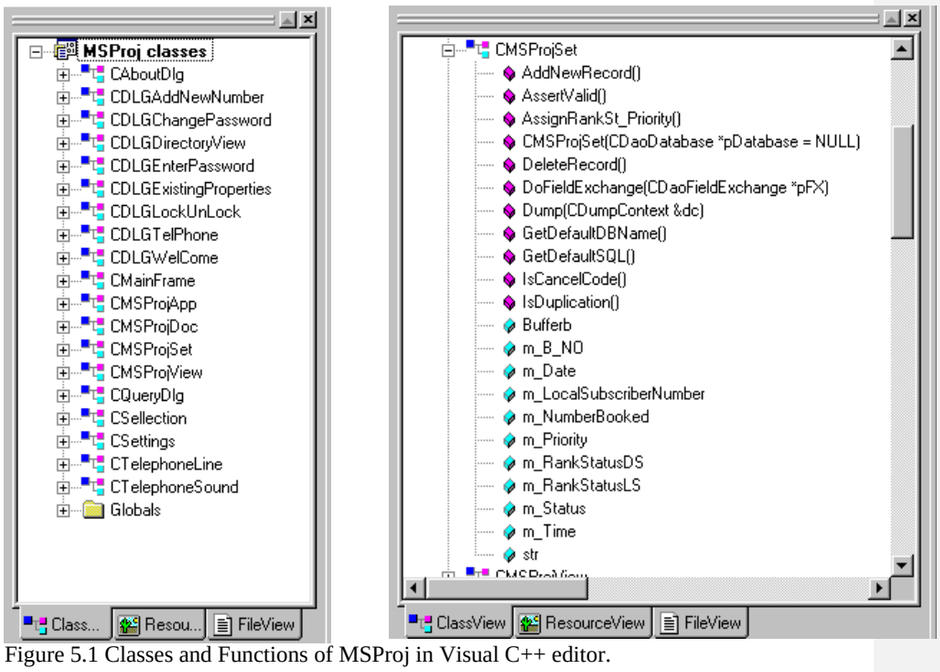


Figure 5.1 Classes and Functions of MSProj in Visual C++ editor.

~~6.1.1~~5.1.1 Class CMSProjSet

The class is associated with a table 'View' of database 'Trunk_Booking'. It contains all the attributes of the table as a member variables which can be accessed by any of the object of this class. It is derived from CDaoRecordSet Class . Its member functions and member variables are describe in the tables 5.1 and 5.2 respectively.

Table 5.1 Member Functions of class CMSProjSet

Visibility	*Name*	*Description*
Public	AddNewRecord	It is a function, which physically add new record to the database. It is called from the View Class whenever new telephone call is receive and enter his number.
Public	IsCancelCode	It is Called within the AddNewRecord function it is one of the levels described above to check whether call is to book a new number or to cancel the existing number.
Public	IsDuplication	It is also called within the AddNewRecord function it is one of the levels described above to check whether call is already in the queue or not.
Public	AssignRankSt_Priority	It is also called within the AddNewRecord function it is one of the levels described above, this function is called to assign the Rank Status to both the number booked and Local number and decides the priority of the number.
Public	DeleteRecord	This function is to delete the unwanted records. It is invoked automatically at the 1st of each month to delete the records of two months back.
Virtual	GetDefaultDBName	This member function is called to determine the name of the database for the Recordset. If a Recordset is created without a pointer to a CDaoDatabase, then this path is used by the Recordset to open the default database. By default, this function returns an empty string.
Virtual	GetDefaultSQL	The framework calls this member function to get the default SQL statement on which the Recordset is based. This might be a table name or an SQL SELECT statement.

Table 5.2 Member Variables of class CMSProjSet.

Visibility	*Data Type*	*Name*	*Description*

Public	CString	m_B_NO	This variable is associated with the field booking number in the database
Public	CString	m_NumberBooked	This variable is associated with the field 'NumberBooked' in the database.
Public	COleDateTime	m_Date	This variable is associated with the field 'Date of Entery' in the database
Public	COleDateTime	m_Time	This variable is associated with the field 'Time of Entery' in the database
Public	CString	m_RankStatusLS	This variable is associated with the field 'Rank Status of the Local Subscriber' in the database
Public	CString	m_RankStatusDS	This variable is associated with the field 'Rank Status of the Distant End Subscriber' in the database
Public	CString	m_Priority	This variable is associated with the field 'Priority of the Call' in the database.
Public	CString	m_Status	This variable is associated with the field 'Status of the Call' in the database

~~6.1.2~~5.1.2 Class CMSProjView

It displays database records in a list. The view is a form like list which is directly connected to a CDaoRecordset object. The view is created from a dialog template resource and displays the fields of the CDaoRecordset object in

the dialog template's controls. It is derived from the CDaoRecordView Class of MFC. Its member functions and member variables are describe in the tables 5.2 and 5.3 respectively Its view at run time is shown in figure 5.2.

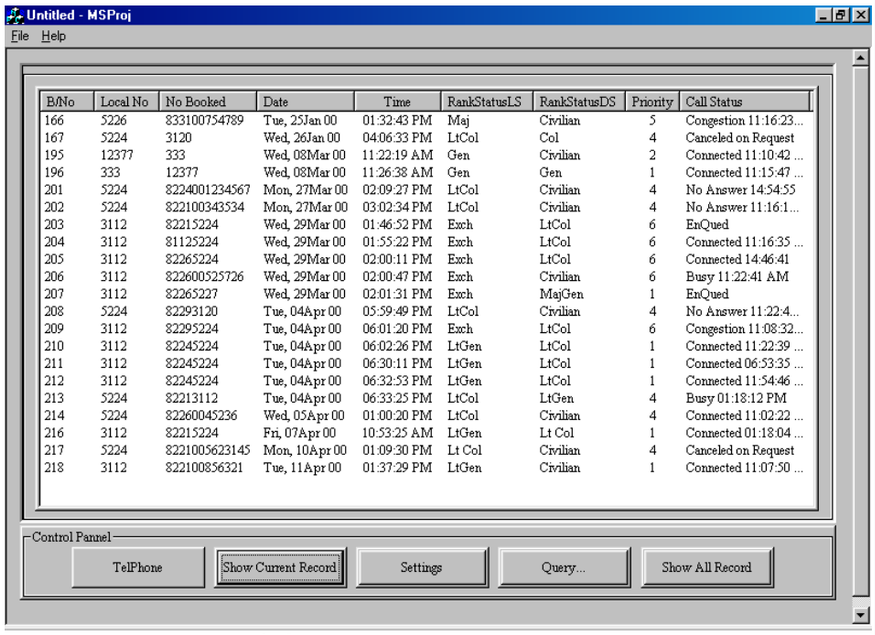


Figure 5.2 Run time view of class MSProjView

Table 5.3 Member Functions of class CMSProjView.

Visibility	*Name*	*Description*
Public	AddColumn	It is a function which add Column headings for the fields shown in the List View. It is called from the Initial Update member function of the dialog.
Public	FillData	It is called from the message generated by 'Show All Record' Button in the main view dialog box. Calling this function fills the screen with the records retrieved from the database.
Public	ShowCurrentRecord	It is called from the message generated by 'Show Current Record' Button in the main view dialog box. Calling this function fills the screen with only the current records retrieved from the database.
Public	ShowQuery	It is called from the message generated by 'Execute Query' Button in the Query dialog box. Calling this function fills the screen with only the selected records retrieved from the database.
Public	ExecuteQuery	This function is called to filter out the query generated, it intelligently work out the query to be further passed on to the Jet Data Base Engine
Virtual	CallAddNewRecord	It is a Raper Function which is called from the Static callback Function of the Class CTelephoneLine, it further call AddNewRecord Function.
Virtual	OnInitialUpdate	Called by the framework after the view is first attached to the document, but before the view is initially displayed. The default implementation of this function calls the OnUpdate member function with no hint information

5.4 Member Variables of class CMSProjView.

Visibility	*Data Type*	*Name*	*Description*
Public	CString	m_B_NO	This variable is associated with the field 'Booking Number' in the database.
Public	CString	m_NumberBooked	This variable is associated with the field 'NumberBooked' in the database.
Public	COleDateTime	m_Date	This variable is associated with the field 'Date of Entery' in the database
Public	COleDateTime	m_Time	This variable is associated with the field 'Time of Entry' in the database
Public	CString	m_RankStatusLS	This variable is associated with the field 'Rank Status of the Local Subscriber' in the database
Public	CString	m_RankStatusDS	This variable is associated with the field 'Rank Status of the Distant End Subscriber' in the database
Public	CString	m_Priority	This variable is associated with the field 'Priority of the Call' in the database.
Public	CString	m_Status	This variable is associated with the field 'Status of the Call' in the database

~~6.1.3~~5.1.3 Class CTelephoneLine

The CTelephoneLine class is designed for the application to establish a TAPI connection. The class, is defined in the TelephoneLine.h file. It is a generic class and not derived from any MFC based class. It directly deals with

Win32 applications. The member functions and member variables are described in more detail in table 5.5 and 5.6.

Table 5.5 Member Functions of class CTelephoneLine.

Visibility	*Name*	*Description*
Public	Create	It Set the necessary properties to null.
Public	InitializeTelephoneLine	The first thing an application must do before it uses any telephony services is to initialize TAPI. This means that the application must establish some way to communicate between itself and TAPI. TAPI uses a callback function to facilitate this communication. The application tells TAPI the address of its callback function when the application makes a call to lineInitialize.The lineInitialize function fills in two values passed to it: a usage handle and the number of line devices available to the application. If the call to lineInitialize is successful, a value of zero is returned. If an error occurs, a negative value is returned.
Public	LineNegotiateAPIVersion	The lineNegotiateAPIVersion function is used to negotiate the API version number to use. It also retrieves the extension identifier supported by the line device, and returns zeros if no extensions are supported.The API version number negotiated is that under which TAPI can operate. If version ranges do not overlap, the application and API or service provider versions are incompatible an error is returned. .
Public	LineCallbackFunc	It is static function it runs asynchronously whenever any message is generated.
Public	OpenTelephoneLine	The lineOpen function opens the line device specified by its device identifier and returns a line handle for the corresponding opened line device. This line handle is used in subsequent operations on the line device.
Virtual	processCallState	It is a function called from LineCallBackFunction it deals with LINECALLSTATE parameters of the line.

Virtual	lineMonitorDigits	The lineMonitorDigits function enables and disables the unbuffered detection of digits received on the call. Each time a digit of the specified digit mode is detected, a message is sent to the application indicating which digit has been detected.
Public	TelephoneLineGetCall Status	The lineGetCallStatus function returns the dynamic status of a call. Call status information includes the current call state, detailed mode information related to the call while in this state (if any), as well as a list of the available API functions the application can invoke on the call while the call is in this state. An application would typically be interested in requesting this information when it receives notification about a call state change by the LINE_CALLSTATE message.
Public	TelephoneLineGetDev Caps	The lineGetDevCaps function queries a specified line device to determine its telephony capabilities. The returned information is valid for all addresses on the line device.
	TelephoneLineGetWaveID	The TelephoneLineGetWaveID function returns a device identifier for the specified device class associated with the selected line, address, or call.
Public	TelephonyShutDown	This function is called before the application is close down.

Table 5.6 Member Variables of class CTelephoneLine

Visibility	*Data Type*	*Name*	*Description*
Public	BOOL	m_bInitialize	This variable is a boolean type which act as a flag and tell whether the line has been initialized properly or not. It is set to false if no line device is

			deducted.
Public	DWORD	m_dwRingCnt	This variable get the number of telephone rings to be counted.
Public	BOOL	m_bLineCallStatusAllocated	This variable is used to check whether space is allocated for structure(LINECALLSTATUS) if not then allocate it other wise free it.
Public	BOOL	m_bLineDevCapsAlloced	This variable is used to check whether space is allocated for structure(LINEDEVCAPS) if not then allocate it other wise free it.
Public	BOOL	m_bLineOpen	This variable is set to true if line is open successfully.
Public	Int	m_DeviceID	This variable takes the Identification of a Device Selected.
Public	DWORD	m_RetApiVersion	This variable gets the API Version returned by 'LineNegotiateAPIVersion' function.

~~6.1.4~~5.1.4 Class CTelephoneSound

The CTelephoneSound class is designed for the application to display the WaveFiles in an audible format which will reply the user during telephonic communication. The class, is defined in the TelephoneSound.h file. The member functions and member variables are described in more detail in table 5.7 and 5.8 respectively. It is a generic class and not derived from any MFC based class. It directly deals with Win32 applications.

Table 5.7 Member Functions of class CTelephoneSound.

Visibility	*Name*	*Description*
Public	Create	It Set the necessary properties to null.
Public	playSound	It is the major function defined in this class. It is called from the CtelephoneLine Class which takes three parameters 'the deviceID','Path of the file tobe

		displayed' and the 'Handle to the window'. It returns Handle to the WaveOut if successful. There are number of functions called within this function..
Public	mmioOpen	The mmioOpen function opens a file for unbuffered or buffered I/O. The file can be a standard file, a memory file, or an element of a custom storage system. The handle returned by mmioOpen is not a standard file handle. It is only use with multimedia file I/O functions.
Public	mmioDescend	The mmioDescend function descends into a chunk of a RIFF file that was opened by using the mmioOpen function. It can also search for a given chunk.
Public	mmioFOURCC	The mmioFOURCC macro converts four characters into a four-character code.
Virtual	mmioRead	The mmioRead function reads a specified number of bytes from a file opened by using the mmioOpen function.
Virtual	mmioAscend	The mmioAscend function ascends out of a chunk in a RIFF file descended into with the mmioDescend function or created with the mmioCreateChunk function.
Public	waveOutOpen	The waveOutOpen function opens the given waveform-audio output device for playback. It uses the waveOutGetNumDevs function to determine the number of waveform-audio output devices present in the system. If the value specified by the uDeviceID parameter is a device identifier, it can vary from zero to one less than the number of devices present. The WAVE_MAPPER constant can also be used as a device identifier.
Public	waveOutPrepareHeader	The waveOutPrepareHeader function prepares a waveform-audio data block for playback. The lpData, dwBufferLength, and dwFlags members of the WAVEHDR structure must be set before calling this

		function (dwFlags must be zero). The dwFlags, dwBufferLength, and dwLoops members of the WAVEHDR structure can change between calls to this function and the waveOutWrite function. (The only flags that can change in this interval for the dwFlags member are WHDR_BEGINLOOP and WHDR_ENDLOOP.) If the change of the size specified by dwBufferLength before the call to waveOutWrite, the new value must be less than the prepared value. Preparing a header that has already been prepared has no effect, and the function returns zero.
	waveOutWrite	The waveOutWrite function sends a data block to the given waveform-audio output device. When the buffer is finished, the WHDR_DONE bit is set in the dwFlags member of the WAVEHDR structure. The buffer must be prepared with the waveOutPrepareHeader function before it is passed to waveOutWrite. Unless the device is paused by calling the waveOutPause function, playback begins when the first data block is sent to the device.
Public	waveOutProc	The waveOutProc function is the callback function used with the waveform-audio output device. The waveOutProc function is a placeholder for the application-defined function name. The address of this function is specified in the callback-address parameter of the waveOutOpen function when a playback is finished this function is called automatically to release the memory blocks.

Table 5.8 Member Variables of class CTelephoneSound.

Visibility	*Data Type*	*Name*	*Description*
Public	Pointer	lpWaveHdr	It is a Pointer to the WAVEHDR structure. It gets the address of the structure returned by the

			calloc function.
Public	Pointer	lpWaveFormat	It is a Pointer to the WAVEFORMATEX structure. It gets the address of the structure returned by the calloc function. Basically it get the size of the format chunk and allocate memory for it.
Public	Pointer	lpWaveData	Allocate and lock the memory for the waveform data.

The Classes described below for this module are all derived from the CDialog Class of MFC they all have almost same and similar number of functions and variables. Therefore, are discussed in a generalized form along with the dialog associated with each class.

~~6.1.5~~5.1.5 Class CDLGAddNewNumber

The class is associated with a DialogBox as shown in figure 5.3 it is used for the addition of new telephone number to the data base. The user will enter the new telephone number in the edit box and select its priority and rank status from

the combo boxes and then press ‘OK’ button, the number will be added to the database and acknowledgement is made to the user for success.

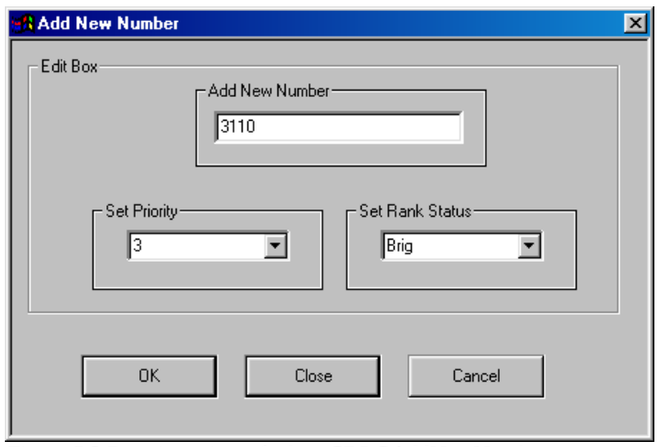


Figure 5.3 Run time view of class CDLGAddNewNumber.

~~6.1.6~~5.1.6 Class CDLGDirectoryView

It is used to view the complete directory along with the rank status and priorities. The user can also delete any number just by the click of mouse on the required number and then press the 'Delete' button. Its view is shown in figure 5.4.

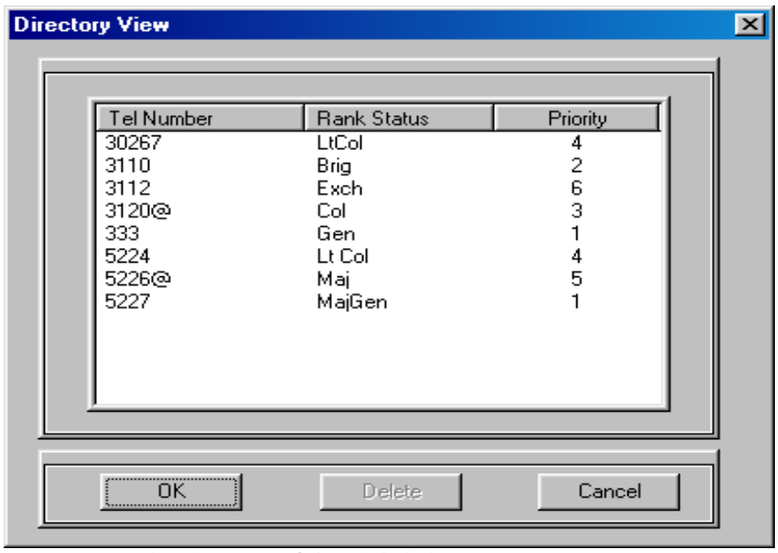


Figure 5.4 Run time view of class CDLGDirectoryView

~~6.1.7~~5.1.7 Class CDLGExistingProperties

It is used to change the priority and rank status of the existing numbers if required, the procedure is the same as for the AddNewNumber Its view is shown in figure 5.5.

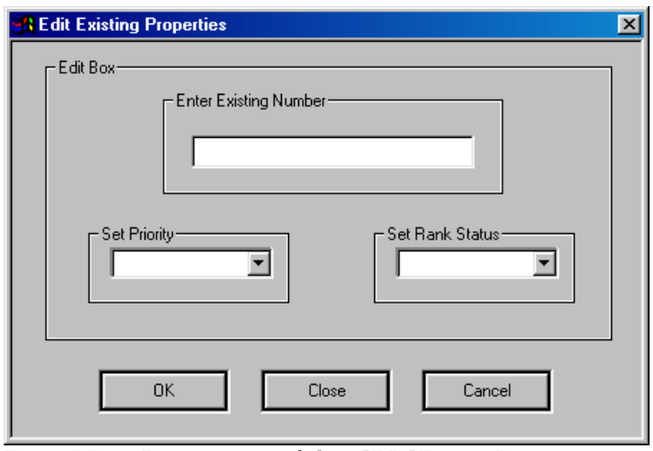


Figure 5.5 Run time view of class CDLGExistingProperties

~~6.1.8~~5.1.8 Class CDLGLockUnLock

It is used to Lock or UnLock the existing telephone number so that no one can book a trunk call from this number. The procedure is very simple just type the number you want to lock in the edit box and press ‘OK” button, the number will be locked, and for unlock just select the number form the combo box which contains the list of all locked numbers and press ‘OK’ button Its view is shown in figure 5.6.

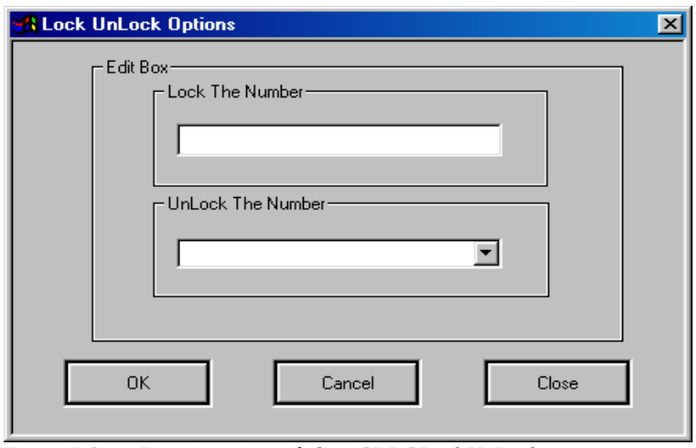


Figure 5.6 Run time view of class CDLGLockUnLock.

~~6.1.9~~5.1.9 Class CDLGChangePassword

This dialog box is used to change the password just by entering new password and confirming it will change the password. Its view is shown in figure 5.7.

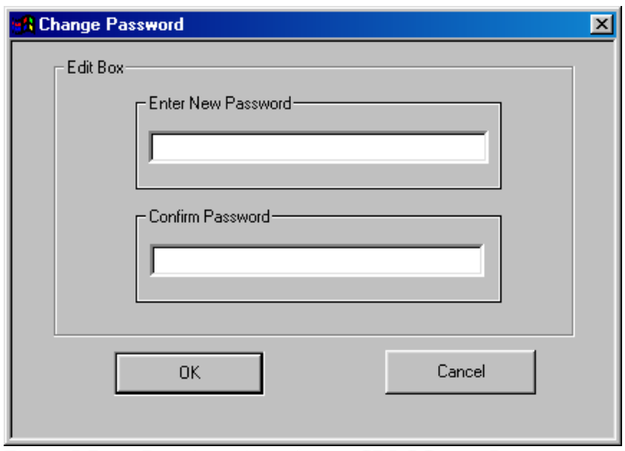


Figure 5.7 Run time view of class CDLGChangePassword.

~~6.1.10~~5.1.10 Class CQueryDlg

It is used to generate wizard oriented query in order to avoid the user to write SQL Query format. The user can have 16 combinations from the following four choices shown in figure 5.8. There are further different choices to select, in the combo box of Call Status and Date.

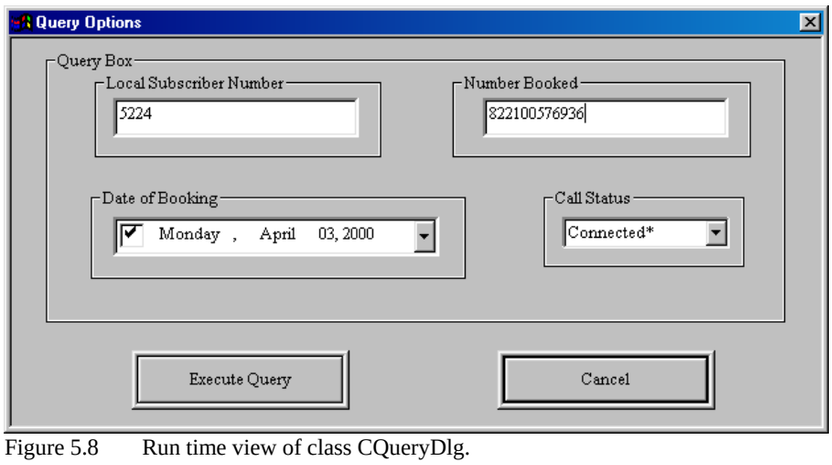


Figure 5.8 Run time view of class CQueryDlg.

The Classes and the member function used in second module are describe in the subsequent paragraphs.

~~6.1.11~~5.1.11 Class CDialerDlg

It is the only user defined class in this module which contain several member functions. It is called at the start of the application. The ControlList used in this dialog based class will flash the selected number to the operator to dial. It is derived from the CDialog Class of the MFC. Its view is shown in figure 5.9. The detail of member functions and member variables are describe in the table 5.9 and 5.10 respectively.

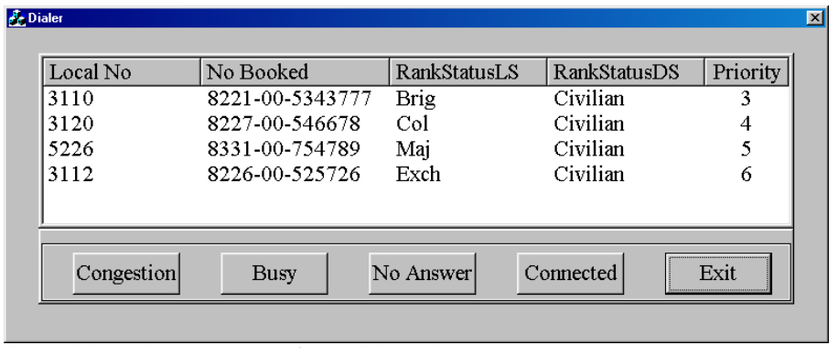


Figure 5.9 Run time view of class CDialerDlg.

After dialing the number the operator has 4 choices according to the situation:

1. Connected.
2. Busy.
3. No Answer.
4. Code Congestion.

Pressing of any button will invoke a routine, which will decide what to write in the database against the number along with the time of service, this will be automatically visible in the 'Call Status' field of main view of project shown in figure 5.2.

Table 5.9 Member Functions of class CDialerDlg

Visibility	*Name*	*Description*
Public	Create	It Set the necessary properties to null.
Public	ScanDbForWaitingCalls	It is the function which is automatically keeps on checking the database for any waiting calls and if it finds any call in the queue it calls the function 'ActivateDialer' to do further which is required.
Public	ActivateDailer	The ActivateDailer function now checks the queue for each priority and picks the number at the top of waiting queue if any and then call the function 'DecideSeniority'.
Public	DecideSeniority	This function will decide the seniority of the number picked by the 'ActivateDialer'.
Public	PrepareToDailNumber	This function is to prepare the number to be displayed on the operator screen.
Public	AddColumn	This function is to add the column headings to the list view.
Public	FillData	This Function is to display the required data to the operator screen.

Table 5.10 Member Variables of class CDialerDlg

Visibility	*Data Type*	*Name*	*Description*
Public	ClistCtrl	m_CtlList	It is an object of the ControlList Class which is used to access the different member function of the class.
Public	CdaoDatabase	db	It is an object of the CDaoDatabase Class which can open the database invoked.
Public	CdaoRecordset	rs	It is an object of the CDaoRecordset Class which can access the tables of the database opened.

Chapter 6

~~7~~6 Conclusion

~~7.1~~6.1 Thesis summary

Corp of Signals plays an important role in the establishment/management of secure and reliable communication network within Pak Army. With the recent break through by acquiring most sophisticated communication infra structure, there is a dire need to optimally/efficiently utilize the resources. Keeping in view this scenario, a system is designed to meet such requirements, that accepts the string entered by the subscriber from telephone pad, then verify its correct format, checks the duplication of number booked, priority of number, and rank status of the subscriber. The systems saves the number booked in the database in the respective logical priority queues and communicate the booking number to the subscriber and the same information is displayed on the screen as well. The system is designed to scan all the logical priority queues automatically after every 5 seconds and the number waiting will be flashed for the operator's attention to connect the call. Two separate modules are designed to handle the booking of number and scanning of the database for the awaiting calls, and are scheduled in such a way that each module will be running in parallel and independent to each other.

Telephonic communication is the most popular means of communication. In Pakistan Army it is one of the major task of Corps of signals to provide the services relating to telephonic communication. Although with the induction of modern equipment the services are much faster and reliable, but still there is room for improvement. Pakistan is economically a poor nation and replacing any existing system abruptly would be a great economical burden. However, if the existing system could be used with slight modification and lesser cost, with additionally faster and reliable results. PASCOM exchanges are inducted in Pakistan Army few years' back. These exchanges have many advanced features in them. However, the trunk booking

handling technique is quite traditional. The booking activity can be replaced in such a way that instead of asking the operator to book a number if the subscriber feed his required number through his telephone to the computer placed at the telephone exchange. By adopting this computerized booking method many ills of the system can be removed.

The working of exchanges are of a traditional type and pays an extra load on the working staff, there is one shift dedicated only to deal with call booking, the job is to entertain the subscriber for booking a call, the procedure for booking is quite lengthy specially when number of calls increases, procedurally he asked the subscriber about his local number, the number he wanted to book, his rank and name, he writes all this information on a chit (known to be a ticket) along with date and time of booking, after that he hand over this ticket to the operator who places it in a queue , Now after connecting the call the operator has to sign the respective ticket and write down the time of connection and reason for not connecting in case call cannot be connected and then hand it over to the exchange supervisor, at the end of the day the supervisor has to maintain the record of all calls in a register for future references, it is a time consuming procedure . Moreover, if any complaint is raised then searching of a specified ticket from that junk is also very time consuming. The concept of the project is to design a system in which the subscriber dials the exchange number and he will then be connected to a computer, removing an intervention of any booking staff. More to say that the computer will play audible sound signal and communicate the booking number.

The major feature of the project is the communication between subscriber and PC, which is dealt through Telephony Application Programming Interface (TAPI), by utilizing Modem and sound card. Database is designed using Microsoft Access.

Various techniques of communication and the differences between the three primary types of telephony hardware for PCs such as Basic data modems, Voice-data modems and Telephony cards were disused in detail.

Telephony Application Programming Interface (TAPI) provides a consistent programming interface for a variety of devices operating on voice grade lines. The devices include modems, FAX modems, voice capable modems, computer-controlled telephone sets, and many more. TAPI provides services for placing outgoing calls, accepting incoming calls, and managing calls and devices.

Telephony cards offer the greatest level of service compatibility. Telephony cards usually support all of the Basic Telephony and all of the Supplemental Telephony services, including phone device control. Most telephony cards also offer multiple lines on a single card. All TAPI services are routed through some type of modem. Sending data over phone lines involves three main steps. If the second modem answers the telephone call, the two modems go through a process of determining if they understand each other called handshaking.

The most advanced level of hardware a programmer can get for TAPI services on a desktop PC is a dedicated telephony card. As with other telephony hardware, telephony cards need an accompanying TAPI driver in order to recognize TAPI calls from user's application. Telephony cards (along with TAPI drivers to match) offer the greatest access to TAPI services. Programmer can support all the Assisted TAPI and Basic TAPI functions along with access to Supplemental TAPI services. Basic data modems support Assisted Telephony services (outbound dialing) and can support only limited inbound call handling. Voice-data modems are capable of supporting the Assisted Telephony and Basic Telephony services and many of the Supplementary services. Telephony cards support all of the Basic Telephony and all of the Supplemental Telephony services, including phone device control.

Data Access Objects, or DAOs, is Microsoft's latest invention in database access technology. Data Access Objects enable the programmers to access and manipulate databases through the Microsoft Jet database engine. Through this engine, programmers can access data in Microsoft Access database files (MDB files). Data Access Objects enable the programmers to access and manipulate database through the Microsoft Jet database engine. Through this engine, programmers can access data in Microsoft Access database files (MDB files). Data Access Object technology is based on OLE (Object Linking and Embading).

Both Data Access Objects (DAO) and Open Database Connectivity (ODBC) are application programming interfaces (APIs) that provides the ability to write applications that are independent of any particular database management system (DBMS). DAO is familiar to database programmers using Microsoft Access or Visual C++. DAO uses the Microsoft Jet database engine to provide a set of data access objects, database objects, tabledef, querydef objects, recordset objects, and others. Microsoft Jet provides a variety of database management services, including data definition, data manipulation, data integrity, and security.

~~7.2~~6.2 Advantages

Following advantages are fore seeable:

Subscriber's end:

1. It totally eliminates the interaction of subscriber with the exchange staff.
4. It gives a satisfaction to the subscriber that the number he booked will be serviced according to his priority.
5. Subscriber has the facility to get the information of his booking number at any time.
4. Subscriber has the facility to cancel his call if need be.

5. Almost eliminates the operator level manipulation regarding management of calls.

Management's end:

1. It totally eliminates the tedious job of exchange supervisor and call booking staff, which includes the management of call booking tickets as well as their record keeping.
2. Now the Operator's job is just dialing of flashed number. He is relieved of tickets handling, their sequencing and endorsement of service time, signatures and reason of not connecting the call in case any call could not be connected.
3. Duplication in call booking is not possible.

Chief Duty Signal Officer's (CDSO) end:

CDSO can enjoy the following privileges, just by the click of mouse:

1. In case of any complaint, the CDSO can settle the issue by accessing the database within no time.
2. Can control the call's booking privilege from any number.
3. Can change the priority and rank status of in use numbers.
6. Can add new numbers in the database as and when required.
7. Can delete a number from the database if no more required.
6. All the settings are protected by a password for security purposes.
7. The manpower relieved by the developed system can be employed elsewhere.

~~7.3~~6.3 Limitations

1. The database designed is for the single PC supported database.
2. The database is being managed centrally by the Chief Duty Signal Officer.
3. Fax Modem doesn't support all features of TAPI.
5. Single channel supported hard ware.

~~7.4~~6.4 Future work

1. ***Speech API*** . For voice recognition.
2. ***TTS(Text to Speech).*** To generate a dynamic wave file instead of prerecorded files.
3. ***Networking***
4. Invoking suitable hardware for multichannel facility.

Bibliography

1. Mastering Visual C++ 6, Michael J.Young, introduction, BPB Publication 1999
2. Mastering Visual C++ 6, Michael J.Young, Chapter 9, Generating a windows GUI program, BPB Publication 1999
3. Microsoft Developer's Network (MSDN), Platform SDK, TSPI, Oct99.
4. Hipson, Peter, Data base developer's guide with Visual C++, chapter 6, part 2, The Microsoft jet database engine, SAMS 1996.
5. Microsoft Developer's Network (MSDN), Platform SDK, TSPI, Oct99.
6. Microsoft Developer's Network (MSDN), Platform SDK, TSPI, Oct99.
7. Sells Charis, Window's Telephony Programming, Chapter 1, Windows telephony overview, PP6, Addison Wesley. Inc. 1998.
8. Microsoft Developer's Network Oct 99.
9. Cathren Recardo ,"Database System Principles, Design and Implementation", Maxwell Macmillan,1990.
10. Young Michael Joe, Mastering Visual C++6.0, chapter 5 Deriving C++ classes.
11. Robert Lafore, Programming for the PC and C++, chapter 16 Object Oriented Programming, SAMS publications.