SPEECH ENABLING INDIAN RAILWAY PASSENGER SERVICES (PNR ENQUIRY)

PROJECT WORK DONE AT
LATTICE BRIDGE INFOTECH Pvt.Ltd,
CHENNAI

PROJECT REPORT

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

M.Sc [APPLIED SCIENCE] SOFTWARE ENGINEERING
OF BHARATHIAR UNIVERSITY, COIMBATORE.

SUBMITTED BY
R.ABISHEK
Reg. No. **9937S0070** [illegible]

UNDER THE GUIDANCE OF

External Guide	Internal guide
Mr. M. SWAMINATHAN **Lattice Bridge InfoTech** **Chennai**	**Mr. K.R BASKARAN** **Asst. Prof. – Dept of CSE - KCT** **Coimbatore**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE – 641 006

OCT 2003 – MARCH 2004

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KUMARAGURU COLLEGE OF TECHNOLOGY
(Affiliated to Bharathiar University)
COIMBATORE – 641 006
OCT 2003 – MARCH 2004

CERTIFICATE

This is to certify that the project entitled

SPEECH ENABLING INIDAN RAILWAY PASSENGER SERVICES (PNR ENQUIRY)

DONE BY

R.ABISHEK
Reg. No. **9937S0070**

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

M.Sc [Applied Science] SOFTWARE ENGINEERING
OF BHARATHIAR UNIVERSITY

[illegible]
Professor and HOD

[illegible]
Internal Guide

Submitted to University Examination held on 29/3/04

[illegible]
Internal Examiner

[illegible]
External Examiner

DECLARATION

I hereby declare that the project entitled "**Speech Enabling Indian Railway Passenger Services (PNR Enquiry)**" submitted to **Bharathiar University**, Coimbatore as the project work of **Master of Science Degree in Software Engineering**, is a record of original work done by me under the supervision and guidance to **Mr. M. Swaminathan [Project Manager]**, LatticeBridge InfoTech Pvt Ltd , **Prof.K.R.Baskaran– Asst.Professor & Course Coordinator [Software Engineering]** , **Kumaraguru College of Technology**, Coimbatore and this project work has not found the basis of the award of any Degree/Diploma/Associate ship /Fellowship or similar title to any candidate of any university.

Place : COIMBATORE

Date : 10-3-2004

[illegible]

R.Abishek
Registration No: 9937S0070
M.Sc [Software Engineering]
Kumaraguru College of Technology

ceBridge
ch Pvt. Ltd.

CERTIFICATE

Date : 28th February, 2004

TO WHOMSOEVER IT MAY CONCERN

This is to certify that Mr. R.Abishek, student of Kumaraguru college of Technology, Coimbatore has successfully completed the project titled " Speech Enabling Indian Railway Passenger Services (PNR Enquiry) in Lattice Bridge Infotech (P) Ltd Chennai.

The duration of the project was from November 2003 to February 2004.The Project was done under the guidance of Ms. S.Srividhya, Project Co-ordinator. The Project attained our objective and his work was found to be satisfactory.

[illegible]
Project Co-ordinator

Authorized Signatory
[illegible]

LATTICE BRIDGE INFOTECH PVT LTD
CHENNAI

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

The satisfaction and Euphoria that accompany the successful completion of any work would be incomplete unless, I mention the name of the people, who made it possible whose constant guidance and encouragement served as a beckon light and crowned my effort with success.

I express my sincere prayers and my heartfelt thanks to my PARENTS and all my friends for their support and loving prayers.

I am greatly thankful to Mr. C. Mohan Ram Managing Director of Lattice Bridge InfoTech (P) Ltd. for permitting me to take part in the project.

I extend my profound gratitude to Dr. K.K Padmanabhan B.Sc. (Eng), M.Tech, Ph.D., Principal, Kumaraguru College of Technology, Coimbatore for providing me an opportunity to do the project works as part of the curriculum.

I express my sincere thanks to my project guides Mr. M. Swaminathan, Ms. S. Srividhya ,Mr. U. Rishi and all other staffs in Lattice Bridge InfoTech for their valuable suggestions and guidance throughout the project.

I am immensely thankful to my guide Mr. K.R Baskaran, Assistant Professor, Dept of Computer science, and course coordinator for M.Sc Software Engineering for his valuable guidance and support throughout my project.

Abishek.R

SYNOPSIS

SYNOPSIS

The project fully goes through the various process that are undertaken to manage the organization and a detail study is done from the data acquired and information gathered. The system is being developed as per the demand of the organization and end user requirement needs

This project entitled "**Speech Enabling Indian Railway Passenger Services (PNR Enquiry)**" is to introduce Speech Recognition System for enquiring the PNR Status of the railway passenger.

The Automatic Speech Recognition (ASR) technology using Telephone media is proposed in the system. With the help of this technology the computer recognizes the input from the user by means of Speech Synthesis. The **SpeechPearl**, a speech recognition engine does the Recognition part in desktop computers with the ability to listen to a user's speech and determines what was spoken. SpeechPearl is a product from Philips Speech Processing Lab, (now part of Scansoft USA). SpeechPearl Open grammar allows an application to restrict what it listens and thus achieves good recognition accuracy and in less response time. The voice is synthesized and converted as Digital data. Data is then compared with the southern railways database using WEBRLL which enable our VOS program with its functions and the relevant information is stored in the ODBC database. The information stored in the database are

accessed using ADO RLL that provides comprehensive database access for Graphical VOS applications which provides the relevant information that the user required.

The system is designed to overcome the problem in the organization without giving rise to ambiguity. The system is designed to operate in a healthy computer environment with the system being user friendly and guiding the user at each step.

CONTENTS

CONTENTS

INTRODUCTION

1.1 Project Overview

The Scope of the project is to introduce speech recognition system for enquiring the PNR status of a railway passenger. The customer should say the digits of the PNR number one by one or dial the number *(works only with DTMF phones, which is by default in major Cities like Chennai, Trichy, Coimbatore etc.).*

The existing system is getting the inputs from the web (www.southernrailways.com) and the IVRS technology. As the Internet is not affordable with many people or access to the Internet is very less in countries like India, the Automatic Speech Recognition (ASR) technology using Telephone media, is proposed in the system. With the help of this technology the computer recognizes the input from the user by means of Speech Synthesis. The voice is synthesized and converted as Digital data. Data is then compared with the southern railways database and the relevant information is given to the user. The proposed solution will be available only in Chennai at the first phase. (*Soon it will be implemented in the remaining places*).

1.2 Organization Profile

Lattice Bridge InfoTech Pvt. Ltd. (LBIT) - pioneer in speech recognition applications in India, is a Global Software Service Provider (GSSP) headquartered in India and delivering IT Application Solutions worldwide. It is founded with the objective of using the power of the Internet & Telephony to provide effective enterprise solutions to its customers. The offering is flexible to meet customer needs and available on traditional deployment or pay-and-use basis from our world-class data center. Our offer also includes System Integration and professional Services, as required by customers backed with 24X7 support.

Lattice Bridge InfoTech (LBIT), is a Chennai based and has a mission **"To make speech the primary user interface for a broad spectrum of customer services"**. We helped Philips to develop Indian English acoustic model to recognize the way Indians speak English and are currently developing models in various other languages, including Hindi and Gujarathi etc.... LBIT has the distinction of launching "first ever" ASR based customer services for a large Public Sector Organization.

Lattice Bridge InfoTech run an Infotainment portal called "Tel-a-phone" service, where Astrology, News, Cricket update etc. is provided with ASR technology in Chennai City. Many more customer friendly services for financial sector, Banks, Logistics companies etc. are under development and would be launched very soon with large prestigious customers.

Lattice Bridge InfoTech are authorized service provider for many railway information services through the Telecom Companies like Touchtel, BSNL, RPG.

SYSTEM STUDY AND ANALYSIS

2.1 Software Requirement Specification

Purpose:

To "Speech Enable PNR Enquiry System of Indian Railways Passenger Amenities".

The main purpose of the project is to introduce Speech Recognition System for enquiring the PNR Status of the railway passenger.

Scope:

The Scope of the project is to introduce speech recognition system for enquiring the PNR status of a railway passenger. The customer should say the digits of the PNR number one by one.

The existing system is getting the inputs from the web (www.southernrailways.com) through the IVRS technology. As the Internet is not affordable with many people, the Automatic Speech Recognition (ASR) technology using Telephone media, is proposed in the system. With the help of this technology the computer recognizes the input from the user by means of Speech Synthesis. The voice is synthesized and converted as Digital data. Data is then compared with the southern railways database and the relevant information is given to the user. The proposed solution will be available only in Chennai at the first phase. (*Soon it will be implemented in the remaining places*).

Definition , Acronyms , Abbreviations :

Definitions :

- Application Programming Interface (API) :

An API is a set of instructions provided by an operation system or device driver to do the limited functionality of the Operating System. Usually the functions are provided as subroutine calls that can be used by programs written in C or another programming language.

- Automatic Call Distributor (ACD) : ACD is a specialized phone system (either Software or Hardware) used for handling many incoming calls and diverting to the called (destination) party. It is used by any company that has large number of incoming calls or sites where many incoming calls are to be answered by one single focal point. (Eg: Big corporate companies Reception).

- Call Progress Analysis (CPA) : CPA is the automated determination by a piece of telecommunications equipment as to the end result of dialing a number. The analysis also involves detecting various call progress tones which are generated by the telephone network as the call is put through.

- CT –ADE : Computer Telephony (CT) Application Development Environment (ADE) is a set of development tools and programming interfaces that help shorten both developer time to market and generate revenue by making it quick and easy to build robust, portable CT applications.

Abbreviations:

CT-ADE	– *Computer Telephony Application Development Environment.*
IVRS	– *Interactive Voice Response System.*
ASR	– *Automatic Speech Recognition.*
ISDN	– *Integrated Services Digital Network.*
POTS	– *Plain Old Telephone System*
PSTN	– *Public Switch Telephone Network.*
AMO	– *Acoustic Modeling*
LMO	– *Language Modeling.*
DTMF	- *Dual Tone Multiple Frequency*

General Description :

Product Perspective :

The tools used for the system are :

- Intel Computer Telephony – Application Development Environment (CT-ADE). (Platform developer for CT applications)
- Speech Pearl 2000. (Speech Recognition Engine)
- MS-SQL Server 2000 RDBMS

Product Functions:

The main function of the product is that to introduce speech recognition system for enquiring the PNR status of a railway passenger. The customer should say the digits of the PNR

Number one by one. With the help of the ASR technology the computer recognizes the input from the user which is the voice of the user. The voice is then compared with the southern railways database and the relevant information is given to the user.

User characteristics:

There is just one person involved in the system. It is the voice of the person which the system recognizes and process the relevant information for the user.

General Constraints:

- As the resonance and frequency – are different and varies from person to person, there are possibilities of misinterpretation at the initial time of implementation.
- Currently the system is not connected to Railways database directly. Input is verified from Website of the System. Hence possibility of delay occurring is expected.
- Railways Computer systems are not available for general public from 23.00 hours to 05.00 hours of next morning. During those times, system functionality is expected to be null.
- The User should say the PNR number one by one. Numbers spelled in Lakhs, thousands, hundreds are not accepted. (*Though system has Natural Language processing skills).*
- The User should use either of the two languages (English or Tamil) only.
- The User should answer to the system's questions correctly.
- User is expected to give the input of all digits in short period.

Specific Requirements :

Functional Requirements :

List of inputs:

The User's voice is the only input for the system. He might be using the voice for the following :

- Language Selection.
- Telling the system the PNR number.
- Repeating the PNR number if the previous said is invalid.
- Continuing or Exit.

Performance Requirements:

Security: Since the system is a Computer Telephony system there are chances of people mishandling the southern railways database system. But the current system has taken necessary measures to avoid these happenings in real time by giving only 3 chances for taking input. If the user fails to give the required input in three attempts, system shall disconnect the call.

Availability : Railways Computer systems are not available for general public from 23.00 hours to 05.00 hours of next morning. During those times, system functionality is expected to be null. Otherwise, System shall be available to any user at all times.

Capacity : Currently two telephone lines are provided and 2 persons can be serviced at a time. However, system could be made serviceable for

30 callers at any given point of time, by connecting an E1 (2MBPS Stream) Digital line.

Response time :

The Response time is usually for 15 seconds between the voice of the user and the reply from the system.

Design Constraints :

Standards Compliance :

Since the system takes the voice of the user as input, at times the system cannot recognize what input the user has said. In those cases the system requests the user to repeat the statement what he has said. If the system is unable to recognize the input for the third time then the system asks the user to speak from a silent area and disconnects the call.

Hardware Limitations :

Right now the system uses a 2 channel process. When the system expands , the whole system will be upgraded to a 30 lines process (E1 in ISDN PRI) or more E1 lines with more E1 cards.

External Interface Requirements :

The external interface requirements required for running the system are a modem , telephone and a telephone line.

User Interfaces , Screen Formats :

The only user interface is going to be Telephone. Since this project is a Computer Telephony system the screen formats are not applicable for the project.

2.2 Existing System

The existing system is getting the inputs from the web (www.southernrailways.com) and user interface is only through Telephone with DTMF Input in the Keypad of telephone.

2.3 Proposed System

As the Internet is not reachable with many people (in India), the Automatic Speech Recognition (ASR) technology is used in the system. As telephone penetration is increasing in our country with many private cellular, basic landline service providers are increasing, reaching the information to the rural people is expected to be high using this technology. With the help of this technology the computer recognizes the input from the user by speech synthesis. The voice is synthesized and converted as Digital data. Data is then compared with the southern railways database and the relevant information is given to the user.

PROGRAMMING ENVIRONMENT

3.1 Hardware Configuration

- A Telephone. (System is configurable for E1 (30) lines)
- Pentium IV Processor. (1.4 GHz)
- 128 MB RAM (256 MB RAM recommended)
- 40 GB Hard disk.
- Dialogic Board (D/4PCI,D/41JCT,D/120JCT or E1 Lines)

3.2 Description of Softwares & Tools Used

- Intel CT-ADE 8.0
- Philips Speech Pearl 2000 (Speech Recognition Engine)
- MS-SQL Server 2000 RDBMS (or any RDBMS)
- Intel Dialogic D/41 JCT- CTI Card.

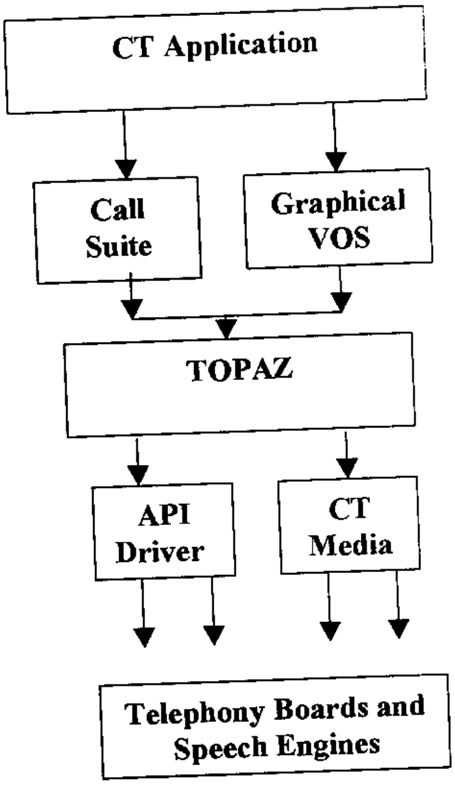
CT Application Development Environment

The Computer Telephony (CT) Application Development Environment (ADE) is a set of development tools and programming interfaces that help shorten both your time to market and your time to revenue by making it quick and easy to build robust, portable CT applications.

The CT ADE mitigates the need to write directly to the API in C or C++. The underlying Topaz architecture eliminates the need for developers to take time learning current and new hardware and API protocols. The Topaz architecture is an abstraction layer that sits on top of an API and performs low-level CT tasks, letting you focus on building your applications without worrying about these low-level operations.

You can access Topaz using either of the two included programming environments: Graphical VOS and CallSuite. The Graphical VOS programming environment, which is described in this help file, is a CT-specific scripting language that includes an optional graphical user interface, (GUI) as well as a debugger. If developer prefer to use a Windows visual development language like Visual Basic, the CallSuite ActiveX controls incorporate directly into the visual programming environment, providing CT-specific development functionality. To learn more about CallSuite, see the CallSuite user's guide, CallSuite.chm, which is included when you install CallSuite from the 40.

Either environment gives you a specialized platform that reduces the repetitive tasks associated with traditional telephony application development, enabling you to stay focused on the essential and profitable aspects of developing your application.



VOS Language

- Robust and Fast

The VOS language is easy to learn, robust, and fast. Specially designed for call processing, there is very little dynamic memory allocation, so live systems are very unlikely to run out of memory. The VOS language is simpler than Visual Basic and much simpler than C or C++. Spawning, multi-call attending are quickly deployable.

- Easier to Learn and Use

VOS was designed to borrow some of the best ideas from the C language, but to avoid many of the problems that make using C difficult for the beginner (syntax) and for the expert (data types, dynamic memory allocation).

For an example of the improved syntax in VOS, if-statements and loops in C have two types: "simple statements" (an expression followed by a semi-colon), and "compound statements" (one or more statements enclosed in curly brackets {...}). In C, this can lead to subtle bugs that are very hard to find. Even experienced C programmers can write a statement like this:

```
for (;;);
  DoThis();
```

The extra semi-colon at the end of the first line means that DoThis() is executed once and is not controlled by the loop. In VOS, a for loop is terminated by an endfor. If you make the same error in the VOS language:

```
for (;;);
   DoThis();
 endfor
```

then it doesn't matter - the extra semi-colon simply creates an empty statement which does nothing, DoThis() is of course still inside the loop. For another example, in C, beginners often make mistakes in using the switch ... case statement because a case "falls through" to the next case unless a break statement is inserted, which is easily forgotten. In VOS, a case is ended by the next case, if you want to "fall through," you must write a goto or use some other technique. VOS does provide some of the same shorthands as C, such as x++ for "add one to x," but if you don't like the shorthands, you don't need to learn them, there is always another way to accomplish your task.

Web RLL

WebRLL provides functions which enable your VOS program to

- Get documents from a Web server
- Get data from active Web pages (e.g. from an e-commerce server)
- Send and receive files via FTP, HTTP

FTP File Transfers

Files can be copied to and from an FTP server using WebPutFile and WebGetFile respectively.

The file on the FTP server is named using a URL that specifies the ftp protocol. By default, anonymous FTP is used, which means that the user name is "anonymous" and no password is given. A login user name (and optional password) can be specified in the URL

The file on the PC is specified using a Windows path name, which may include drive and/or path name components. By default, the file is assumed to be in the current drive and current directory when VOS was started.

Waiting for a Web Transfer

The Web transfer functions WebPutFile, WebGetFile and WebGetData are all "asynchronous" — that is, they return immediately rather than blocking the VOS task. The WebWait function is used to wait for the transfer to complete and return the result of the transfer. The application can test whether the transfer has completed without blocking by using WebBusy.

This design is convenient for interactive telephony applications that need to make Web transactions in response to the telephone user's input. Since there may be a delay of several seconds or more before the transfer completes, the application may want to play messages during the transfer so that the caller is not listening to silence.

.IPF, Indexed Prompt Files

An Indexed Prompt File (IPF) is a collection of VOX files combined into a single file. Each VOX file stored within an IPF is called a prompt. An IPF begins with a header which includes an index listing all of the included prompts.

IPFs can be created with the Mkipf utility. The directory of an IPF can be viewed with the Dmpipf utility, and an IPF can be "exploded" into its component prompts by using the Expipf utility.

IPFs offer the following advantages:

- There is less delay required to start playing a prompt in an IPF than a given VOX file. This is because VOS already has the file open. You will hear faster responses from your system if you use IPF prompts.
- Phrases can be played which consist of a series of prompts from IPF files, these sound smoother than playing one .vox file after another.

- There are fewer files to administer, for example when distributing your application, if several VOX files are combined into an IPF.

There are also disadvantages of IPFs:

- You can only play from, not record to, an IPF prompt when your VOS application is running. IPFs are therefore suitable only for prompts that rarely or never change in your system since you would need to stop your application and copy a new IPF to change a prompt.
- IPFs may become very large. This may be a problem in distributing your application since the IPF may not fit on a single diskette.
- You may have an additional step of building the IPF when you make a change to the application. When your application is changing rapidly under development, it may be easier to deal with individual VOX files.

The IPF format used by VOS is compatible with that used by the VBASE40 speech file editor distributed by Dialogic, and by other voice file editors such as VFEDIT.

IPF Include File

The IPF include file is a TZP file that lets you specify which Indexed Prompt Files are opened when Topaz starts. These IPFs can be played directly, and they can also be used by the Topaz International Phrases.

For each IPF, you need to set

- An IPF name.
- The name and path to the .ipf file.

- Properties of the audio formatting of the IPF, including coding, sample rate, and a channel count.

IPFCount

The IPFCount Profile ID specifies how many IPFs are to be defined. Remember that in the include file IPFs are numbered starting with 0:

```
\IPFCount=3
\IPFs[0]\...
\IPFs[1]\...
\IPFs[2]\ ...
```

IPFName

The IPF name is used by functions or vocabulary entries to specify which IPF file contains a desired prompt.

IPFFile

This Profile ID specifies the name and path to the IPF file.

AudioFormatCoding

The audio format coding specifies whether the file was recorded with A-law or mu-law companding. Valid values for AudioFormatCoding are Alaw or muLaw.

AudioFormatSamplesPerSecond

This Profile ID specifies the sampling rate, measured in samples per second (Hz). VOX files use 6 kHz or 8 kHz, so valid values for AudioFormatSamplesPerSecond would be 6000 or 8000.

AudioFormatBitsPerSample

A sample is a binary number specifying the amplitude (loudness) of the sound at the instant a measurement was taken. Sample sizes will generally be 4-, 8- or 16-bit. Valid values for AudioFormatBitsPerSample are 4, 8, or 16.

AudioFormatChannelCount

Files recorded in stereo have a channel count of 2, and mono files have a channel count of 1.

Example

```
\IPFCount=2

\IPFs[0]\IPFName=English
\IPFs[0]\IPFFile=c:\vos\ipfs\Eng_us_f.ipf
\IPFs[0]\AudioFormatCoding=muLaw
\IPFs[0]\AudioFormatSamplesPerSecond=8000
\IPFs[0]\AudioFormatBitsPerSample=8
\IPFs[0]\AudioFormatChannelCount=1



\IPFs[1]\IPFName=Spanish
\IPFs[1]\IPFFile=c:\vos\ipfs\Span_sa_f.ipf
\IPFs[1]\AudioFormatCoding=muLaw
\IPFs[1]\AudioFormatSamplesPerSecond=8000
\IPFs[1]\AudioFormatBitsPerSample=8
\IPFs[1]\AudioFormatChannelCount=1
```

ADO RLL

The ADO RLL is a VOS Runtime Link Library that provides comprehensive database access for Graphical VOS applications. Projects created in the Graphical VOS IDE require the ADO RLL.

The ADO RLL provides VOS applications with access to the Microsoft ActiveX Data Objects (ADO). ADO supports a wide range of database file types, including SQL servers, ODBC data sources and many traditional PC file formats such as dBase, FoxPro, Paradox, Excel, Lotus and others.

All access to data happens through Connections. The sequence is typically:

Open a Connection.
Get the Recordset you want.
Specify a record.
Read or write fields in the record.

Connections

Without exception, the first object that you will need is a Connection object, which you create with ABOpPub and ABOpPrv. A Connection object represents the session between your application and a given data provider. Often, API documentation refers to objects with this kind of functionality as a "database," but "Connection" is a more accurate term.

The "Connection string" is used to specify which data provider to connect to, and to specify provider-specific information (such as your user-name, or the database file that the provider should connect to). See Connection Strings for more information.

Recordsets

Once you have made a Connection you will probably want to start working with the data you now have access to. To do so you will need to create a Recordset object. A Recordset is, as the name implies, a set of records. A "table" is also a set of records, but the difference lies in location: tables exist within databases, while Recordsets exist within applications and are usually a subset of one of a database's tables.

Recordsets are created from Connections. The link between a Recordset and its Connection is very important. For instance, certain capabilities of a Recordset may be made possible or not possible depending on the parameters used when the Connection was opened. For instance, if the Connection was opened as read-only, any Recordsets opened on that Connection also will be read-only. In addition, the ADO RLL does not permit a Connection to be closed unless any Recordsets that were opened have also been closed.

Opening Recordsets with the ADO RLL

There are two ways to open a Recordset in the ADO RLL. First, you can use ASOpPub or ASOpPrv. These functions take a "source" string which tells the provider which records to put into the new Recordset. Typically, this string is either the name of a table from which to retrieve all records, or an SQL query that selects a subset of records to retrieve from a table. The other parameters of ASOpPub and ASOpPrv provide much control over the Recordset that you create. For instance, you have control over the type of cursor used, plus the level of access (read/write/exclusive, etc) you and others have to the data.

The downside of using ASOpPub and ASOpPrv is that they do not provide the ability to use special database features called stored procedures or parameterized queries. In order to use either of those features, you will need to use ASPubCmd or ASPrvCmd. These functions create Recordsets through the use of a special object called a Command. Each Connection object has a single Command object associated with it (though the object is not used when ASOpPub/ASOpPrv are used). The ABSetCmd and ABPrepared functions set the Command's objects properties.

To further complicate things, each Command object has what is called a "collection" (think of it as a list or array) of objects called Parameters. (Actually, underneath the scenes there are two distinct classes: a "Parameters" object represents a set of "Parameter" objects.) There are various functions by which you can set and retrieve information about these Parameters.

The downside to using ASPubCmd and ASPrvCmd is that the Recordsets created are always read-only and forward-only. Thus, ASPubCmd and ASPrvCmd should only be used if you require the use of stored procedures or parameterized queries.

Accessing Data in a Recordset

Once you have a Recordset you can start to access the data in it. ADO only allows access to one record at a time within a Recordset. The Recordset "cursor" points to the record you can access. The ADO RLL provides a number of functions for moving this cursor around--see Scrolling, the Cursor and the Current Record for more on these functions. There are also a few functions for determining where the cursor is located, but in general it is harder to determine where the cursor is than to put it where you want it. Part of this is due to the fact that the database and/or Recordset may have other users accessing it, so that for instance the record that was the first is now the second, or perhaps has even been deleted by another user! You will want to familiarize yourself with a feature supported by some providers called Bookmarks. These allow you to uniquely identify a record, and come back to it whenever you wish.

Each record in a Recordset is made up of one or more Fields, which are commonly represented as the columns in an table. The ADO RLL hides most of the nitty-gritty of fields from you and instead provides you with a few functions that allow you to get data from and set data in a field. One thing you will want to be aware of is how data types are handled. VOS has only one type of data: strings. In contrast, ADO supported databases can have dozens of different types of data. It is beneficial to be aware that the ADO RLL has to convert strings to these other data types and back each time you set data to or read data from a database.

ADORLL.INC

ADORLL.INC is provided with the ADO RLL for your convenience. This file contains constants that match most of the ADO enumerated values that can be passed to ADO RLL functions.

What is Topaz ?

Topaz is collection of software modules. The core component is a thin layer of code that mediates between your programming commands (like

VoiceBocx1.PlayFile in VoiceBocx or MediaPlayFile in VOS) and the underlying Resource API. For example, if you invoke the VOS function TrunkAnswerCall, the Topaz layer determines which trunk interface is being used, whether it is legal to make this call, and then which API function to use: dx_sethook (R4 analog), dt_setsig (R4 T-1), cc_Answer (R4 PRI), and so on.

Topaz works in a very thin, efficient layer which uses very little memory and adds only a tiny incremental CPU cost compared with calling native APIs directly from C or C++ code. For example, Topaz avoids expensive operations (loops, table searches...) except in one-time operations which are performed at machine boot or application load.

Topaz Introduction

The most important feature of Topaz is API transparency. This means that the same set of functions works under all supported telephony APIs (Dialogic R4, etc.) and all supported trunk types (analog loop-start, T-1 E&M, E-1/R2, PRI...).

Functions that work with are also simple, easy to learn and consistent. Compared with C programming, this means that typical application functionality (making or receiving calls, playing menus and prompts, and so on) is substantially easier to create and maintain. Topaz relieves the developer from dealing with many chores which native APIs require, such as:

- Driver initialization.
- Opening and closing resources.
- Complex API function signatures including handles, bit-fields, pointers and structures.
- Asynchronous event handling (call-backs, messages etc.).

- Determining the installed hardware and trunk configuration.

Additionally, because Topaz is API transparent, you avoid locking yourself into a commitment to a particular API. Unlike a C/C++ program, the Topaz application will be portable to different APIs and trunk types as the industry evolves and your end-user needs change.

Topaz is "under the hood" of all CT ADE development tools, including CallSuite™ ActiveX controls and VOS™. To use VOS or CallSuite, you must set up the Topaz.ini File and Topaz Profile.

Topaz Profile :

The Topaz Profile is a database that is quite similar to the Windows registry--it's a tree of directories and files that stores the following information:

- Details of all installed hardware Resources, as determined by the Resource Scanner.
- User-supplied hardware configuration information that cannot be determined by the Resource Scanner.
- User-configurable options, such as the default language for speaking values (English, Spanish...).

Running VOS applications treat the Profile as read-only. The Profile should be fully initialized before these applications are started.

No dynamically changing information, such as the current state of a Resource, is stored in the Topaz Profile.

Creating and Configuring the Topaz Profile :

Unlike the Windows Registry, user application code has no direct access to the Profile and cannot create new entry names. The Profile is for internal use by Topaz only.

The Topaz Profile is created by running the TopazProfile program after completing the configuration steps below.

By default the Topaz Profile is installed on the host machine's system drive in:

C:\Program Files\Parity Software\Common\Topaz\Profile

In order for Topaz to run, the Topaz.ini [Profile] section must specify the path to the Topaz Profile.

You configure certain Resource parameters in the Topaz Profile by making appropriate entries in the Topaz Configuration File (Topaz.ini) or in Profile include files.

Voice Recognition

Voice recognition (VR) technology (also known as speech recognition or automatic speech recognition) reliably recognizes certain human speech, such as discrete numbers and short commands, or continuous strings of numbers, like a credit card number.

Speaker-independent VR can recognize a limited group of words (usually numbers and short commands) from any caller. Speaker-dependent VR can identify a large vocabulary of commands from a specific speaker. Speaker-dependent VR is popular in password-controlled systems and hands-free work environments.

Elements of Speech Recognition

Language resources are essential elements of speech recognition applications, as they contain computerized information about the sound of spoken utterances.

Their quality influences the performance of speech recognition systems. The language resources offered by Philips Speech Processing cover a wide variety of different recognition tasks. To suit geographical and cultural needs, Philips language resources are available in multiple languages.

There are the following language resources:

- lexicons,
- acoustic models,
- language models, and
- auto-transcription resources.

Lexicons, acoustic models and auto-transcription resources are the basic elements for creating SpeechMania or Speech Pearl applications. These items are delivered by Philips Speech Processing and are specially trained and adapted to their specific recognition tasks. Language models always depend on a certain application and are created within the process of creating or training an application. The details of language resources are described in the following sections.

Lexicon

A lexicon reflects the common pronunciation of words in certain languages.

It contains words and their transcriptions determining the pronunciation. One word can have several transcriptions, indicating its base pronunciation and the pronunciation variants.

Transcriptions are written down in phonetic symbols, so-called SAMPA symbols.These are computer-readable phonetic symbols corresponding to the common IPA symbols which are used in dictionaries.

Acoustic Model

In an acoustic model (AMO) speech is being split into smallest units of spoken sounds. An AMO contains information about the acoustics of these sounds.

As different languages contain different phonetic sounds, an AMO is always unique for a certain language. In order to provide acoustic models for various languages we collect speech data from all over the world. Philips Speech Processing is a member of a world-wide network of partners collecting speech data, for example ELRA, LDC, SpeechDat and other industrial consortia.

Acoustic models are created and trained by collecting many different speech sounds to obtain all varieties of human speech from lots of speakers varying in age, sex, and dialects. As more and more people are using cellular phones our AMOs are adapted for both fixed network and cellular phone applications.

Language Model

A language model is used to train an existing application to ensure the success of the online system's recognition components.

Language models contain statistical information about which words are used mostly by the callers of an application. The system always accepts the hypothesis with the highest probability as a result of the recognition process.

Language models are always application dependent. To train a language model for a certain application it is necessary to collect authentic and representative speech data, covering various speaker habits such as different vocabulary and the way of expressing oneself.

Collecting authentic and representative speech data means that the callers say phrases the system has to understand when it is deployed. If you build e.g. a train time table application you first develop a prototype application that volunteering people have to call. During this data collection the callers have to utter city names so that the language model later knows which city names are requested mostly by the callers.

Using the collected data, the system learns to match the acoustic input to the correct textual representation. The more speech data is collected, the better the language model can be trained. A good language model is essential for high recognition performance.

Auto-Transcription Resources

Orthographic Auto-Transcription (addWord):

Orthographic auto-transcription resources are necessary for the orthographic automatic transcription (OAT) of words which can be performed by SpeechXpert and the addWord feature provided by SpeechPearl.

OAT is performed when creating the application lexicon for words that are neither contained in the related background lexicon nor in the user lexicon.

An OAT resource is made up of the following files: "<language>.lrf" and "<language>_phon.txt". The latter is the phoneme inventory which contains all phonemes of the respective language in SAMPA format.

Acoustic Auto-Transcription (learnWord)

The learnWord feature enables your application to use the patented Philips Speech Processing technology for learnWord applications – the real time acoustic automatic transcription (AAT) of spoken names. With this technology you can build personal call assistant applications.
An AAT resource is made up of the following files: "<language>_learnword.lex" and "<language>_learnword.amo".

Philips Speech Pearl 2000:

Speech Pearl 2000 is a software-only speech recognition and natural language understanding engine that is well-suited for a wide variety of telephony applications such as service automation, voice mail control and many others. It provides voice platforms with a "say-what-you-mean" interface for telephony speech recognition applications. It is not limited to isolated word recognition, one item at time recognition, or to keywords-only recognition. To achieve this, Speech Pearl 2000 supports speaker independent speech recognition and understanding for small and large vocabularies, and it is optimized for continuous digit recognition. The Speech Pearl 2000 recognizer also supports tone recognition for Asian languages, e.g. Mandarin.

Philips Speech Pearl Grammars :

The Philips Speech Grammar Format (PSGF) is based on the Java Speech Grammar Format (JSGF). PSGF defines a rule grammar, also known as regular grammar. A rule grammar determines which caller utterances the SpeechPearl recognition module shall be able to understand at a certain dialog state. Natural language understanding means the capability to recognize all kinds of utterances spoken by the caller, without imposing any restriction to the way he formulates his request, so the caller has the full freedom of expression.In SpeechPearl 2000 semantic information can be enclosed in a grammar. In this way it is possible to interpret the recognition results on a high level of performance.

Open Grammar Mode

For SpeechPearl 2000 a new grammar mode has been developed – the open grammar mode. The open grammar mode is a powerful extension of the closed grammar mode used in SpeechPearl 99. As open grammars are no longer based on full sentences but only on meaningful concepts, they reduce the effort in grammar design enormously. The result is an application where the caller has the full freedom of expression without being restricted to any given word sequence within a sentence.

A concept is a non-terminal that defines a certain word or word sequence forming a sense unit within a sentence. When processing a recognized sentence, the speech understanding module only searches for grammar matching concepts instead of covering each word in the sentence.

Example:

```
grammar pizza;
language enu_standard;
conceptset
  { <size> , <topping> }

<size> = small
        | medium
        | large
        ;
<topping> = ham
              | mushrooms
              | extra cheese
        ;
```

This grammar contains two concepts, namely <size> and <topping>. With an open grammar like this, caller utterances like the following could be understood easily:
"Give me a small pizza with ham, please."
"I want extra cheese on my pizza." (This sentence uses only one concept.)
"Hi, ahem, I'll take a pizza with mushrooms, a large one, please."

Closed Grammar Mode

In closed grammars the start rule has to cover the complete caller utterance, from the first to the last recognized word in the sentence.

Example:

```
grammar pizza;
language enu_standard;
startrule
      {
       I'd like a <size> pizza with <topping>
      }
<size> = small
        | medium
        | large
        ;
<topping> = ham
              | mushrooms
              | extra cheese
        ;
```

If you have a closed grammar like this, the caller is restricted to the word sequence as defined in the start rule.If the caller says a sentence like "I'd like a large pizza with extra cheese." then the utterance will be understood because it is covered by the grammar.

But if the caller says something like "I'll take a small pizza with ham, please." Then the utterance is not covered by the grammar and will not be understood.

Open Grammar Mode versus Closed Grammar Mode

There is a simple but essential difference between open and closed grammars.Even if these two grammar formats syntactically only differ in one header declaration - namely "startrule" for closed or "conceptset" for open grammars - they are processed in totally different ways by the speech understanding module of SpeechPearl 2000.

The advantage of parsing concepts instead of full sentences is that callers have the full freedom of expression and do not need to adhere to any restricted word sequence. This approach is computationally far simpler, so that the system can work faster and more reliable.

MS-SQL Server 2000:

SQL Server 2000 is the latest and most powerful version of Microsoft's data warehousing and relational database management system. This new release is tightly integrated with Windows 2000 and offers more support for XML, as well as improved Analysis Services for OLAP and data mining.

Professional SQL Server 2000 provides a comprehensive guide to programming with SQL Server 2000, from a complete tutorial on Transact-SQL to an in-depth discussion of new features, such as indexed views, user-defined functions, and the wealth of new SQL Server features to support XML.

SYSTEM DESIGN

4.1 Input Design

There are two modes through which the user gives the input. One is the Speech and the other is the DTMF (Telephone Keypad). During the speech , the caller is expected to speak only the relevant input. For example only the digits has to be spoken digit by digit. (*Digits such as double , triple are not expected as these words has different phonemes with varied demographics such as India).* During DTMF inputs if user presses any key other than numeric system shall give error message.

Inputs such as confirmation (*Yes* , *No*) are expected to be recognized with synonyms words also. For example for the word "Yes" , user may give identical words such as *Yeah* , *Right* , *Correct* , *Go Ahead*, *Yeap etc*. The same scenario in DTMF input mode is done away with 1 for Yes and 2 for No.

List of inputs:

The User's voice is the only input for the system. He might be using the voice for the following:

- Language Selection.
- Telling the system the PNR number.
- Telling the PNR number if the previous said is invalid.
- Confirming the given PNR number
- Requesting the system to repeat the status.
- Continue or Exit.

4.2 Database Design

Passenger Details :

Table name : pass_stat

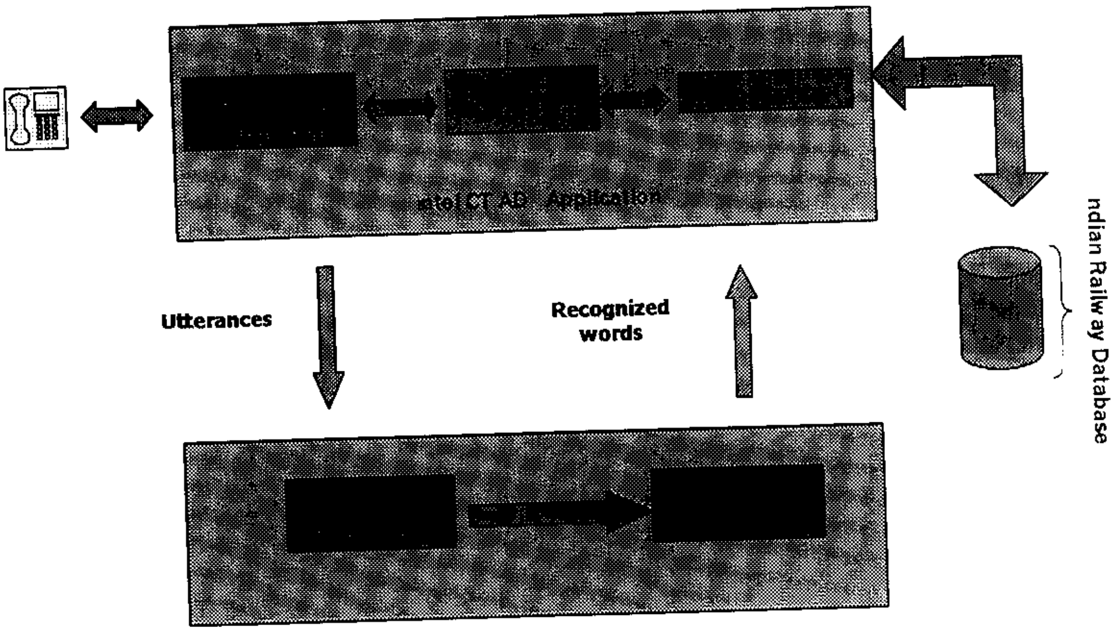
Field Name	Data Type	Size
Pnr_no	Number	10 (primary key)
Train_no	Number	5
Train_name	Char	25
Doj	Date	Short date
From	Char	25
To	Char	25
Reserved_upto	Char	25
Current_stat	char	10
Class	Char	10
Coach_no	Char	4
Berth_no	Number	2
Seat_no	Number	2

Call Log Details :

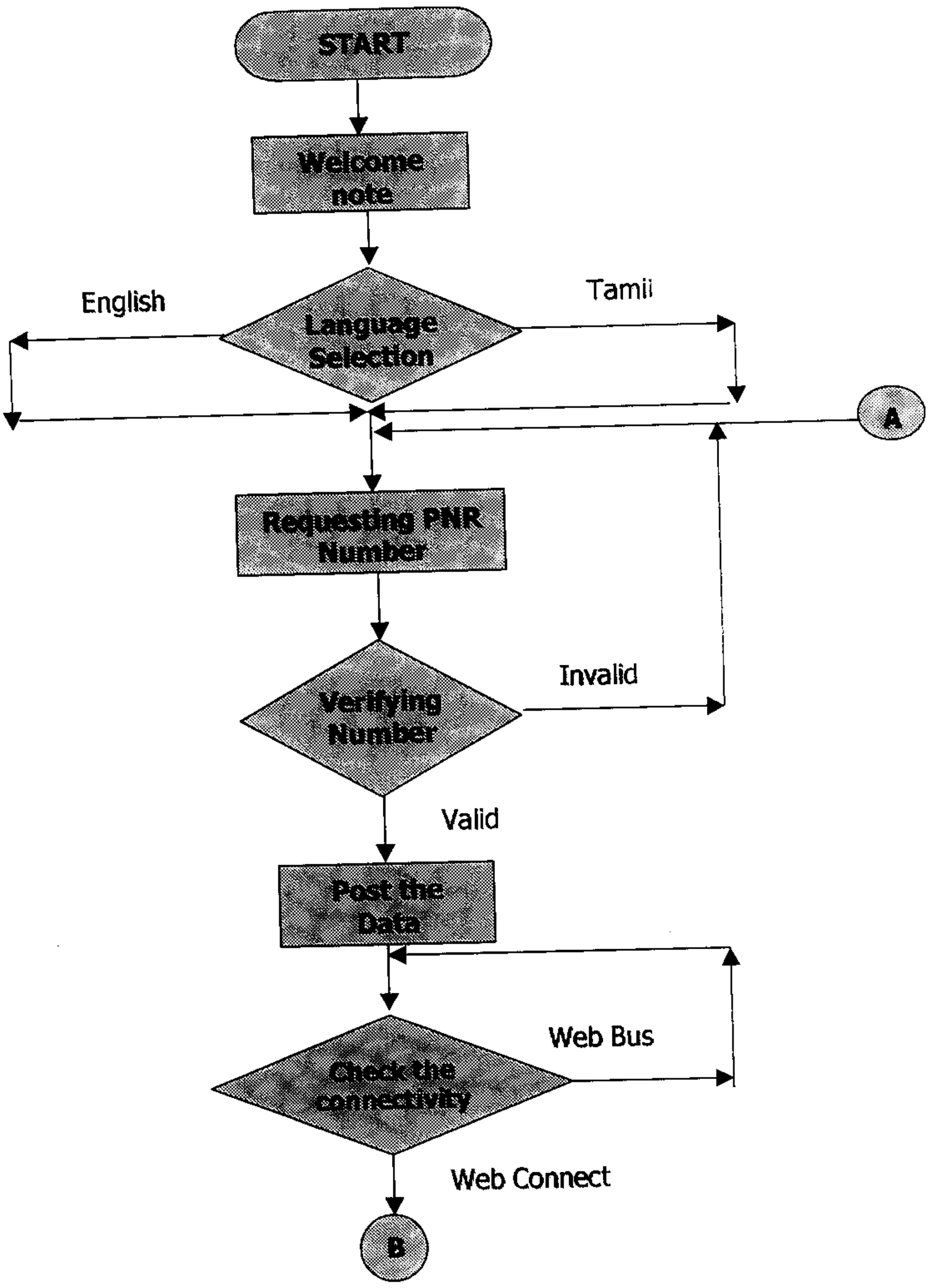
Table Name : call_log

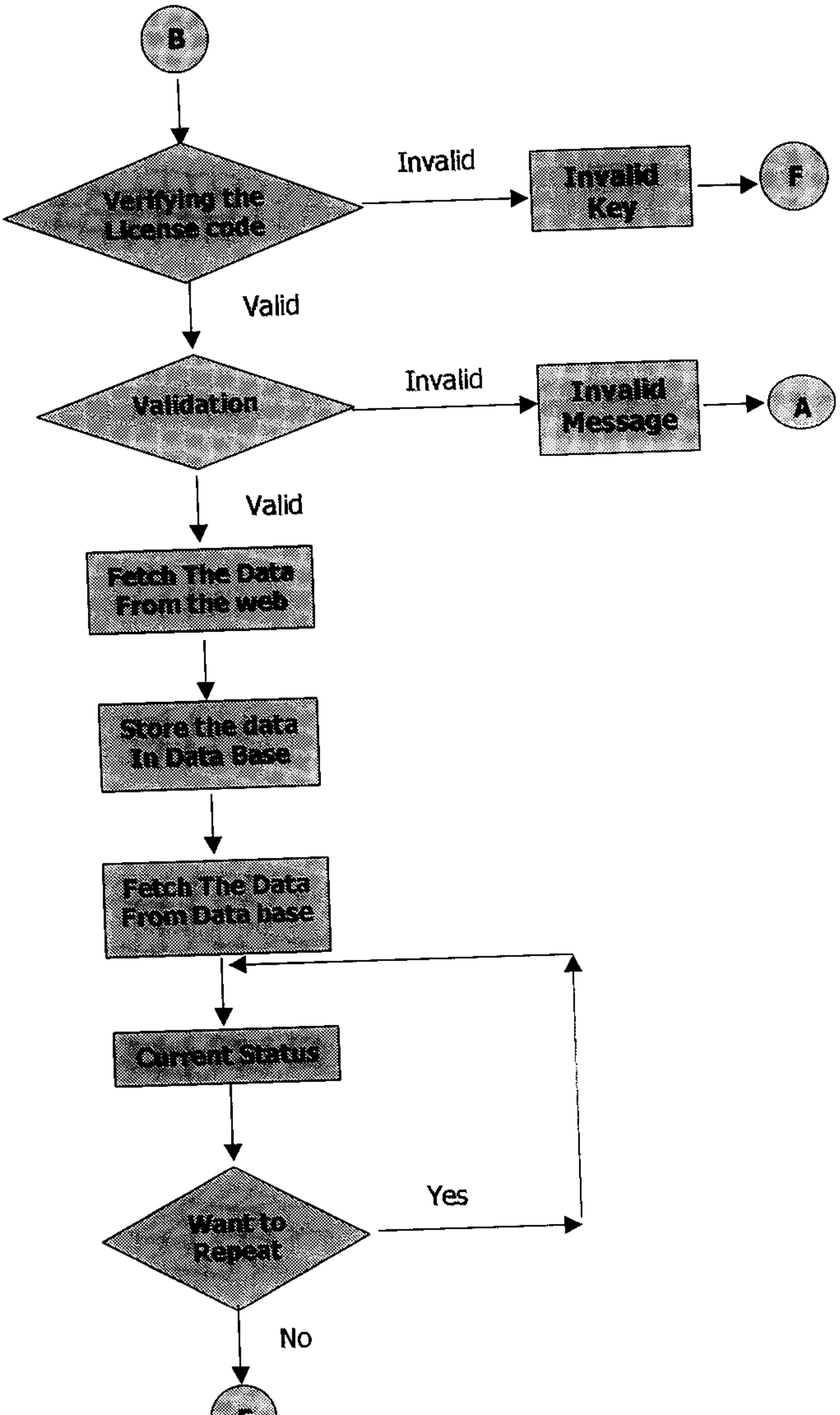
Field Name	Data type	Size
Caller_id	Number	12
Caller_date	Date	Short_date
Caller_time	Time	Short time
Pnr_number	Number	10
Result	Char	10
Call_end	Time	Short time
Call_duration	Time	Short time

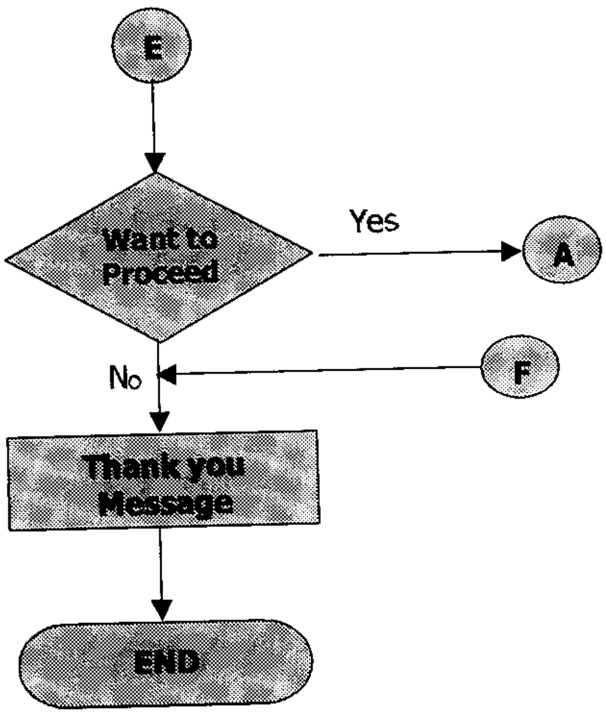
4.3 Process Design



4.4 Flowchart




B
Verifying the License code
Invalid
Invalid Key
F
Valid
Validation
Invalid
Invalid Message
A
Valid
Fetch The Data From the web
Store the data In Data Base
Fetch The Data From Data base
Current Status
Want to Repeat
Yes
No


E
Want to Proceed
Yes
A
No
F
Thank you Message
END

SYSTEM IMPLEMENTATION AND TESTING

5.1 System Implementation

Implementation is the stage when theoretical design is turned in to a working model. This stage is considered as a critical stage in achieving a successful new system. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve change over, training of user staff and evaluation of change over methods.

The processes conducted during the implementation stage are,

- Testing of developed modules with sample data.
- Correction of errors.
- Testing the system to meet user requirements.
- New files with actual data had been created.
- Changes were made according to user's suggestions.

Users were given training for the new system.

5.2 System Testing

The important step in the development process of a system is testing the application to check it whether it is running properly and efficiently. Since this application is a real time system, the ideal test data should also be of real time.

Testing Strategies

- Unit Testing:

This is not applicable in this project as most of the CTI applications are bundled as one solution containing interface as telephone and output also as telephone. However individual modules such as databases connections, network connectivity, and user interface were carried as mentioned below:

- CTI Call Interface Testing:

Every time when the computer is switched on or different user is logging dialogic control manager (DCM) is activated with services of windows2000 operating system. Without DCM being active dialogic services cannot be enabled. DCM is added in the services of windows 2000 OS with automatic option.

This testing is carried out with logging of different users of the PC. Also whenever a new Topaz profile is created DCM's service were active and number of Dialogic boards, Number of Channels were updated. Hence confirmed the Test result is positive.

- Network connectivity:

In this project, as railway systems & computers were accessed via Serial Communication port (RS-232C), its test result significance is more important. Serial Ports are activated with STX Command of RS-232C which is accessed by SOCKET.RLL (or DLL in WIN32 terms). Tests for sending, receiving, time-out, validity of the data send & received are tested.

The computer connected to the Railway Systems were located in the same premises and hence the time-out error were very minimal or we can call it as nil.

On multi-threading part of Serial communication, the team did not explore much as the licenses given by Railways for testing is only one port. But this cannot be treated as flaw or disadvantage of the system. To increase the throughput of the system, PC may be upgraded with more serial port cards (available in the market) or change the method of connecting as RS-485 based communication. By this more ports at Railways end could be accessed by one PC at our end.

At the outset, on result of this test could be considered as improvement in changing to RS-485 is necessary.

- Database Access Testing:

Database access in CT-ADE is bundled as option of accessing with ADO functions of Microsoft Database Drivers. Database is used for recording the caller details such as "Caller id, date of calling, time of conversation beginning and ending, type of call and nature of call ending (such as Caller hanging the phone or system disconnecting the calls).

Number of calls made and number of records in the table were verified. Also the type of call ending (hang-up & disconnected too confirmed to match the expected results).

Security

System is protected against any vulnerability of hacking / crash/ malicious functioning as the interface to the user is only telephony.

Verification

Verification refers to the set of activities that ensure that the software correctly implements a specific function, imposed at the start of that phase. Testing activity focuses on verifying the correct implementation of business requirement and customer requirement.

Validation

Validation refers to the set of activities that ensure the software that has been built is traceable to customer requirements. Validation includes activities like Code-walkthrough to ensure that the software conforms to set standards.

Software Testing

Software Testing is a systematic activity aimed to uncover errors in a software program with respect to its specification to fulfill stated requirements.

- ### Integration Testing

Integration testing refers to the testing in which software units of an application are combined and tested for evaluating the interaction between them. Black box test case design are most prevalent during integration, though white box testing techniques like Control flow graphing and Execution tracing are also carried out.

Inter module and inter product integration issues are the prime focus areas here. We concentrate on the application's business rules and ensure they are validated across different modules.

- System Testing

Testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. Software once validated for meeting functional requirements must be verified for proper interface with other system elements like hardware, databases and people. System Testing verifies that all these system elements mesh properly and the software achieves overall function / performance.

We carry out Product audit and acceptance, performance testing as a part of system testing.

Functional Testing :

➢ Black Box Testing

Those testing methods that need functional understanding of 'what' a software unit is supposed to perform rather than 'how' forms a part of Functional Testing.

Business rule validations through sample data in a test sequence come under this type of testing.

- Regression Testing

    - Regression Testing refers to the selective re-testing of a system or component to verify that modifications have not caused unintended effects and the system component still conforms to the specified requirements.

- Stress Testing

    - Stress testing is a type of system testing that aims at confronting the system with varied levels of abnormal situations in terms of consumption of computer resources like quantity, frequency or volume.

- Performance Testing

    - Performance testing is a type of system testing that aims to determine whether a system meets the performance requirements within the physical limitation of the computing environment of the system.

CONCLUSION

CONCLUSION

The "SPEECH ENABLING INDIAN RAILWAYS PASSENGER SERVICES" has been developed and the user should say or dial the PNR number to know the current status. The testing process is going on. The speech recognition system developed is waiting for the approval from the railways.

The goals that have been achieved by the developed systems are:

1. It simplifies operations.
2. It requires less time and eliminates the manual work..
3. User friendly and could get the required informations.
4. Flexible for further enhancements and maintenance.

SCOPE FOR FUTURE DEVELOPMENT

SCOPE FOR FUTURE DEVELOPMENT

The future scope of the project is to expand the current application all over Tamil Nadu and adding the Hindi language to the existing Tamil and English. Later the Call back facility is also to be implemented in the system.

BIBLIOGRAPHY

Books:

1. Intel Computer Telephony – Application Development Environment. – Manual , 1999
2. Philips speech Pearl 2000. – Manual , 1998
3. SAMPA Notation.

Websites:

www.southernrailways.com

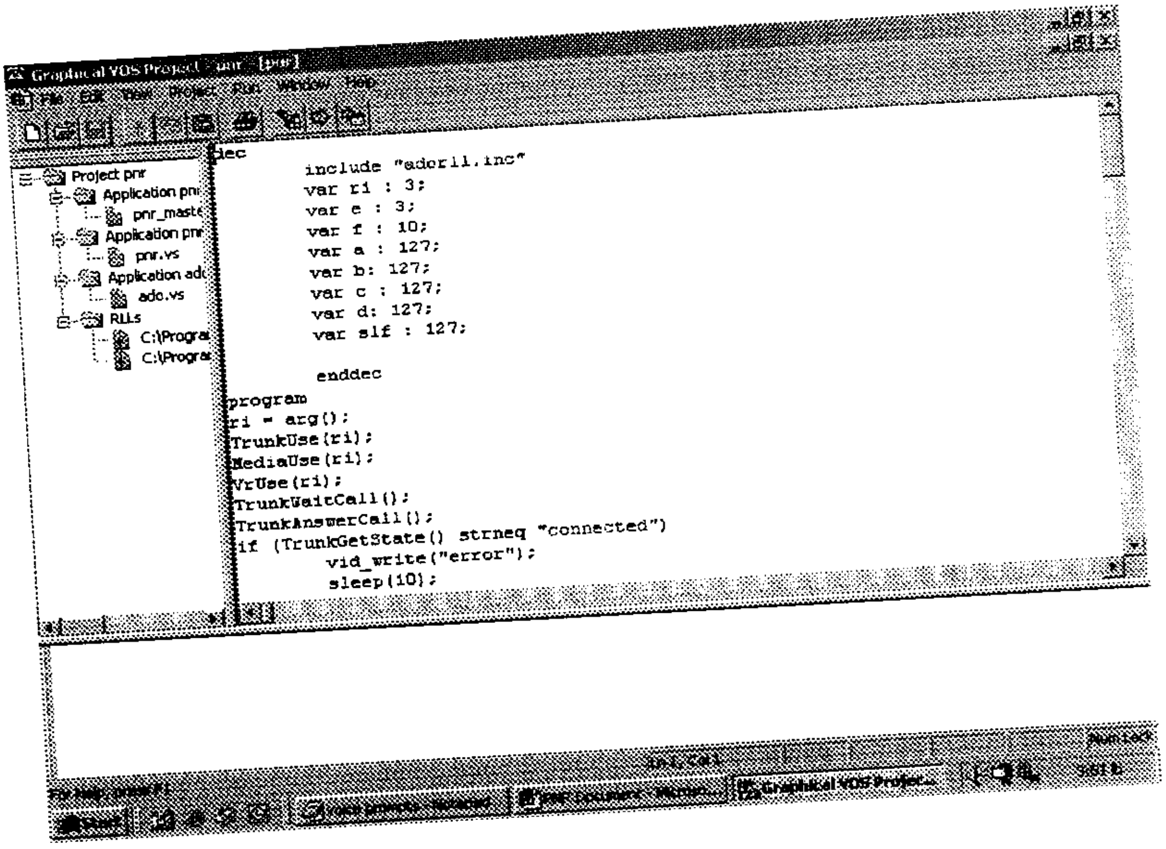
http://www.indianrail.gov.in

www.lbinfotech.com

www.scansoft.com

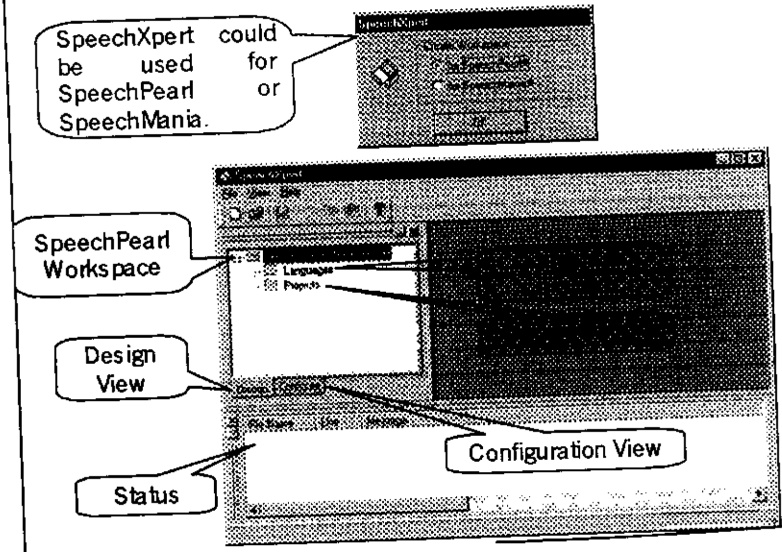
http://microsoft.com/speech

9.1 SAMPLE SCREENS



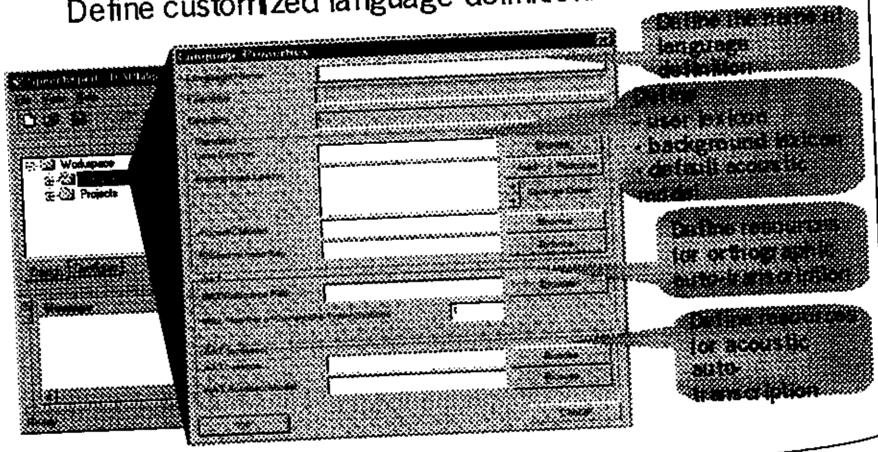
```
dec
        include "adorll.inc"
        var ri : 3;
        var e : 3;
        var f : 10;
        var a : 127;
        var b: 127;
        var c : 127;
        var d: 127;
        var slf : 127;

        enddec
program
ri = arg();
TrunkUse(ri);
MediaUse(ri);
VrUse(ri);
TrunkWaitCall();
TrunkAnswerCall();
if (TrunkGetState() strneq "connected")
        vid_write("error");
        sleep(10);
```

SpeechXpert



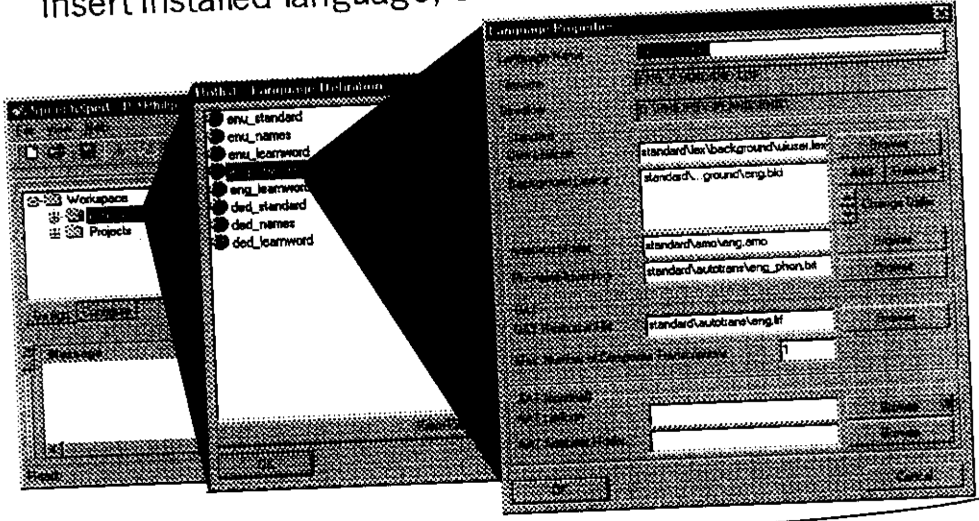
Language Properties

Define customized language definition:


Define the name of language definition
Define:
- user lexicon
- background lexicon
- default acoustic model
Define resources for ortographic auto-transcription
Define resources for acoustic auto-transcription

Language Properties

Insert installed language, select from a hot list or from file:

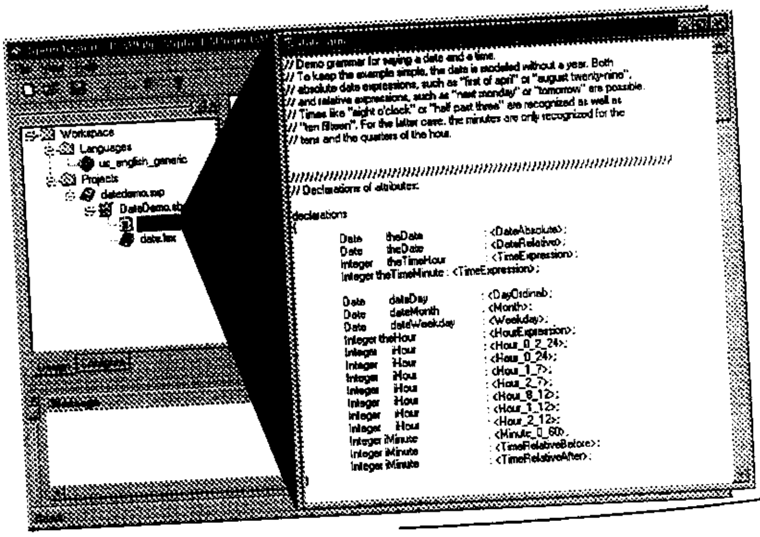


Workspace
Projects

enu_standard
enu_names
enu_learnword
eng_learnword
ded_standard
ded_names
ded_learnword

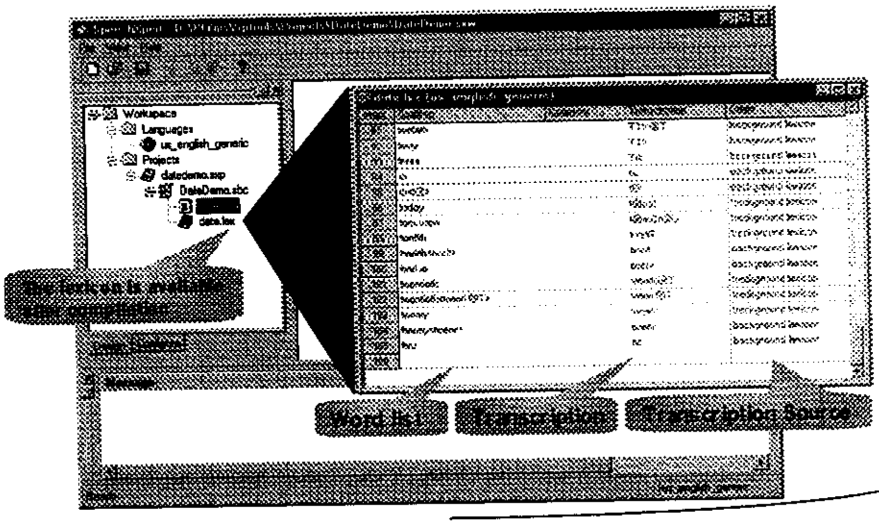
standard\lex\background\uiuser.lex
standard\...ground\eng.bld
standard\amo\eng.amo
standard\autotrans\eng_phon.txt
standard\autotrans\eng.trf
1

SpeechBlock - Grammar Editor

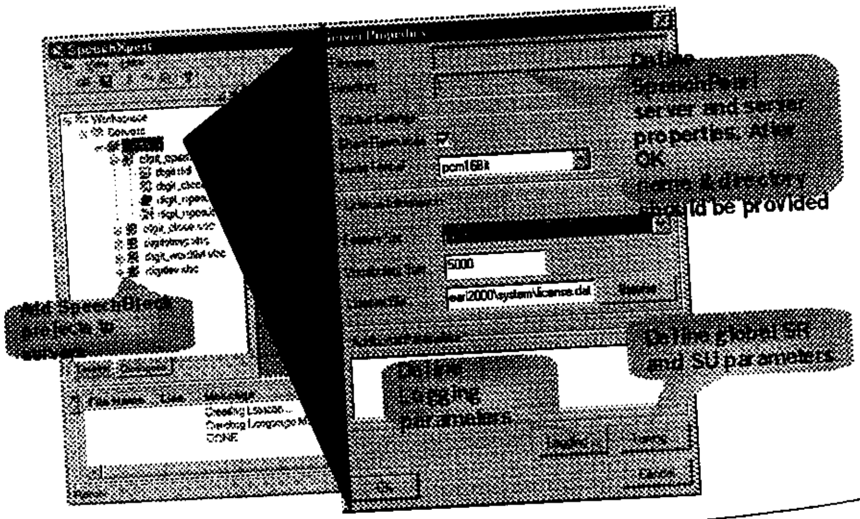


```
// Demo grammar for saying a date and a time.
// To keep the example simple, the date is modeled without a year. Both
// absolute date expressions, such as "first of april" or "august twenty-nine",
// and relative expressions, such as "next monday" or "tomorrow" are possible.
// Times like "eight o'clock" or "half past three" are recognized as well as
// "ten fifteen". For the latter case, the minutes are only recognized for the
// tens and the quarters of the hour.


/////////////////////////////////////////////////////////////////////
// Declarations of attributes:

declarations
{
        Date      theDate              : <DateAbsolute>;
        Date      theDate              : <DateRelative>;
        Integer   theTimeHour          : <TimeExpression>;
        Integer theTimeMinute : <TimeExpression>;

        Date      dateDay              : <DayOrdinal>;
        Date      dateMonth            : <Month>;
        Date      dateWeekday          : <Weekday>;
        Integer theHour                : <HourExpression>;
        Integer   iHour                : <Hour_0_2_24>;
        Integer   iHour                : <Hour_0_24>;
        Integer   iHour                : <Hour_1_7>;
        Integer   iHour                : <Hour_2_7>;
        Integer   iHour                : <Hour_8_12>;
        Integer   iHour                : <Hour_1_12>;
        Integer   iHour                : <Hour_2_12>;
        Integer iMinute                : <Minute_0_60>;
        Integer iMinute                : <TimeRelativeBefore>;
        Integer iMinute                : <TimeRelativeAfter>;
}
```

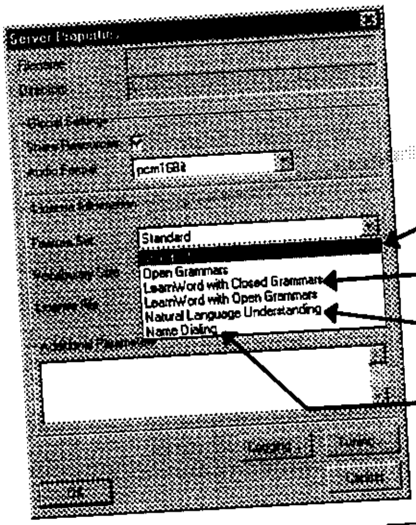
SpeechBlock - Lexicon Editor



Define SpeechPearl Server



Select Feature Set


Server Properties
Open Grammars
LearnWord with Closed Grammars
LearnWord with Open Grammars
Natural Language Understanding
Name Dialing
OK

Audio Format: pcm16Bit

Feature Set: Standard

Select Feature which should match your license.

Closed Grammars

Closed Grammars using learnWord

Open Grammars
= Natural Language Understanding*

LearnWord with Open Grammars
= Name Dialing *

*Compatible names (old)

9.2 SAMPLE CODE

Coding for DTMF mode

```
dec
        include "adorll.inc"
        var ri : 3;
        var con :127;
        var rs : 127;
        var db : 127;
        var nr : 127;
        var i  : 127;
        var A[1..20] :25;
        var train_no : 25;
        var booking_stat : 15;
        var lenStr : 25;
        var digit1 : 25;
        var y_n : 5;
        var coach: 5;
        var st1 : 15;
        var st2 : 15;
        var st3 : 15;
        var st4 : 15;
        var berthno : 4;
        var coachno : 127;
        var seatno : 4;
        var coach1 : 127;
        var pnr : 10;
        var berth1 :127;
        var temp  : 10;
```

```
      var ret : 2;
enddec
program
       ri = arg();
       TrunkUse(ri);
       ret=MediaUse(ri);
       vid_write("mediause" &ret);
       vid_write("wait for call");
      TrunkWaitCall();
       TrunkAnswerCall();
       if (TrunkGetState() strneq "Connected")
       vid_write("Trunk not connected");
       TrunkDisconnect();
       else
       vid_write("connected");
       ret = MediaPlayFile ("C:\aaa\prompts\WELCOME.WAV");
       vid_write("mediaplayfile" &ret);

 #open a db
              con = "DSN=pnr;UID=;PWD=;DATABASE =pnr;";
              db = ABOpPub("My",con,0);
              if (db eq 0)
                     vid_write("db is open");
              else
                     vid_write("db is not open");
              endif

  # open a recordset
   rs=ASOpPrv("My","select*from
  pass_status",adOpenKeyset,adLockPessimistic,adCmdText,adUseServer);
        if (rs eq 0)
```

```
            vid_write("rs is open");
           else
                   vid_write("rs is not open");
           endif
           nr = ASNrRecs();
            vid_write("No. of records are "&nr);
            temp = 0;
repeat:
             temp = temp + 1;
            if (temp eq 3)
             MediaPlayFile ("C:\aaa\prompts\thanku.WAV");
            TrunkDisconnect();
             exit(0);
             endif
             MediaPlayFile("C:\aaa\prompts\ur_pnr.wav");
             sleep(10);
        vid_write("enter the 10 digit number");
       MediaWaitDigits(10,15,2,"*");
       pnr = MediaGetDigitBuffer();
        lenStr = length(pnr);
        pnrvalue(pnr);
        MediaPlayFile("C:\aaa\prompts\sayy_n.wav");
        MediaWaitDigits(1,3,1,"*");
        y_n = MediaGetDigitBuffer();
        if (y_n eq 1)
               if (lenStr eq 10)
                      ASMvTop();
                      temp  = 0;
    #get data
               for (i = 1;i <= nr; i++)
               A[i] = ASGet("pnr_no");
        if (A[i] eq pnr)
        vid_write("pnr found");
        train_no = ASGet("T_no");
        booking_stat = ASGet("Booking_status");
        break;
        else
        ASMvNxt();
        temp = 1 + temp;
        if (temp eq nr)
        MediaPlayFile ("C:\aaa\prompts\invalid.wav");
        goto repeat ;
       endif
    endif
    endfor
repeatagain:

     MediaPlayFile("c:\aaa\prompts\stat_now.wav");
     Status(booking_stat);
    stringtoDigit(train_no);
     MediaPlayFile("c:\aaa\prompts\repeat.wav");
 say_yes:
     MediaPlayFile("C:\aaa\prompts\sayy_n.wav");
     MediaWaitDigits(1,3,1,"*");
     y_n = MediaGetDigitBuffer();
     if (y_n eq 1)
     goto repeatagain;
     else
     if (y_n eq 2)
           MediaPlayFile("c:\aaa\prompts\continue.wav");
          MediaPlayFile("C:\aaa\prompts\sayy_n.wav");
           MediaWaitDigits(1,5,1,"*");
```

```
            y_n = MediaGetDigitBuffer();
           if(y_n eq 1)
          goto repeat;
      else

MediaPlayFile("c:\aaa\prompts\thanku.wav");
       TrunkDisconnect();
       endif
    else
        goto say_yes;
        endif
      endif
        else
       MediaPlayFile ("C:\aaa\prompts\invalid.wav");
        goto repeat ;
        endif
        else
        goto repeat;
        endif
        endif
        TrunkDisconnect();
    endprogram
    onhangup
            MediaPlayFile ("C:\aaa\prompts\thanku.WAV");
            TrunkDisconnect();
    end

     func stringtoDigit(digit)
            lenStr = length(digit);
           MediaSetLanguage("English_India");
           MediaPlayFile("C:\aaa\prompts\and.wav");
    MediaPlayFile("C:\aaa\prompts\your.wav");
    vid_write("Your train no is " );
    for(i =1; i <= lenStr; i++)
      digit1 = substr(digit,i,1);
      vid_write(digit1);
              digit1 = digit1 &".wav";
              MediaPlayFile("C:\aaa\prompts\"&digit1);
       endfor
return digit;
endfunc

func pnrvalue(digit)
        lenStr = length(digit);
       MediaPlayFile("C:\aaa\prompts\isur_pnr.wav");
       vid_write("your pnr is " );
       for(i =1; i <= lenStr; i++)
         digit1 = substr(digit,i,1);
         vid_write(digit1);
                 digit1 = digit1 &".wav";
                 MediaPlayFile("C:\aaa\prompts\"&digit1);
          endfor
   return digit;
   endfunc


   func Status(stat)
   vid_write(stat);
           st1 = ctoi(stat);
           st2 = ctoi ("confirmed");
           st3 = ctoi ("RAC");
           st4 = ctoi ("wait_list");
```

```
if (st1 streq st2)
        MediaPlayFile("c:\aaa\prompts\confirm.wav");
        MediaPlayFile("c:\aaa\prompts\coachno.wav");
        vid_write("Your coach number is ");
        coachno = ASGet("coach_no1");
        coachstat(coachno);
        coachno = ASGet("coach_no");
        coachnostat(coachno);
        MediaPlayFile("c:\aaa\prompts\berthno.wav");
        vid_write("Your berth number is ");
        berthno = ASGet("seatno");
        berthnostat(berthno);
  else
        if (st1 streq st3)
                MediaPlayFile("c:\aaa\prompts\rac.wav");
                MediaPlayFile("c:\aaa\prompts\coachno.wav");
                coachno = ASGet("coach_no1");
                coachstat(coachno);
                coachno = ASGet("coach_no");
                coachnostat(coachno);
                MediaPlayFile("c:\aaa\prompts\seatno.wav");
                vid_write("Your seat number is ");
                berthno = ASGet("seatno");
                berthnostat(berthno);
          else
                if (st1 streq st4)
                        MediaPlayFile("c:\aaa\prompts\waitlist.wav");
                        berthno = ASGet("seatno");
                        vid_write("Your position is ");
                        berthnostat(berthno);
                endif
```

```
              endif
        endif
        return stat;
endfunc

func coachstat(coach)
        lenStr = length(coach);
        for(i =1; i<=lenStr; i++)
        coachno = substr(coach,i,1);
        vid_write(coachno);
        coachno = coachno &".wav";
               MediaPlayFile("C:\aaa\prompts\"&coachno);
        endfor
        return coach;
  endfunc

  func coachnostat(coach)
          lenStr = length(coach);
          if (lenStr eq 1)
                 for(i =1; i<=lenStr; i++)
                        coach1 = substr(coach,i,1);
                        vid_write(coach1);
                 endfor
                 coach1 = coach1 &".wav";
                 MediaPlayFile("C:\aaa\prompts\"&coach1);
          else
          for(i =1; i<=lenStr; i= i + 2)
                        coach1 = substr(coach,i,2);
                        vid_write(coach1);
                 endfor
                 coach1=coach1 &".wav";
```

```
                MediaPlayFile ("C:\aaa\prompts\" &coach1);
        endif
return coach;
endfunc

func berthnostat(berth)
        lenStr = length(berth);
        if (lenStr eq 1)
                for(i =1; i<=lenStr; i++)
                        berth1= substr(berth,i,1);
                        vid_write(berth1);
                endfor
                berth1 = berth1 &".wav";
                MediaPlayFile ("C:\aaa\prompts\" &berth1);
        else
                for(i =1; i<=lenStr; i = i +2)
                        berth1 = substr(berth,i,2);
                        vid_write(berth1);
                endfor
                berth1 = berth1 &".wav";
                MediaPlayFile ("C:\aaa\prompts\" &berth1);
        endif
return berth;
endfunc
```

Coding for Speech mode

```
dec
        include "adorll.inc"
        var ri : 3;
        var con :127;
        var confidence: 16;
        var con1: 127;
        var rs : 127;
        var db : 127;
        var nr : 127;
        var i  : 127;
        var A[1..20] :25;
        var train_no : 25;
        var booking_stat : 15;
        var int : 3;
        var lenStr : 25;
        var digit1 : 25;
        var stat : 25;
        var digit :10;
        var pn : 10;
        var st1 : 15;
        var st2 : 15;
        var st3 : 15;
        var st4 : 15;
        var berthno : 4;
        var coachno : 4;
        var seatno : 4;
        var value :3;
        var retval_VR : 1;
        var temp : 10;
```

```
     var temp1 : 10;
     var retval : 1;
     enddec
program
      ri = arg();
      TrunkUse(ri);
      MediaUse(ri);
      VrUse(ri);
      retval_VR = VrStartSession();
      vid_write("wait for call");
      value=TrunkWaitCall();
      vid_write("Trunkwaitcall"&value);
      value=TrunkAnswerCall();
      vid_write("Trunkanswercall"&value);
  if (TrunkGetState() strneq "Connected")
              vid_write("Trunk not connected");
              TrunkDisconnect();

  else
        vid_write("connected");
         vid_write("vrsession" & retval_VR);
         MediaPlayFile ("C:\aaa\prompts\WELCOME.WAV");


   #open a db
         con = "DSN=pnr;UID=;PWD=;DATABASE =pnr;";
         db = ABOpPub("My",con,0);
         if (db eq 0)
               vid_write("db is open");

         else
               vid_write("db is not open");

         endif
```

```
#open a recordset
rs=ASOpPrv("My","select*frompass_status",adOpenKeyset,
adLockPessimistic,adCmdText,adUseServer);
    if (rs eq 0)
            vid_write("rs is open");

    else
            vid_write("rs is not open");

    endif


 #used to find no of records
        nr = ASNrRecs();
        vid_write("No. of records are "&nr);
 retvalue:
        MediaSetLanguage("English_India");
        Retval=VrPlayAndRecogDigits("C:\aaa\prompts\ur_pnr.vox",10,10,
 "",0,0,"vox,8000,8,mulaw");
 vid_write ("Ret val for VrplayandRecog :"&retval);
 retval = VrGetHypoCount();
 vid_write ("No: of Hypothesis :"&retval);


         if (VrGetHypoCount() < 1)
        MediaPlayFile ("C:\aaa\prompts\invalid.wav");
        goto retvalue;
         endif
          con1 = VrGetHypoStr(1);
        vid_write("PNR Number :" &con1);
         confidence = VrGetHypoScore(1);
         vid_write("confidence value: " &confidence);
               if (confidence < 50)
          vid_write("confidence value: " &confidence);
           goto retvalue;
        endif
        # f = VrGetHypoStr(1);
         vid_write("PNR Number :" &f);
         MediaPlayFile ("c:\aaa\prompts\isur_pnr.wav");
         if (confidence > 50)
        VrPlayAndRecogYesNo("C:\aaa\prompts\sayy_n.wav");
        if (VrGetHypoCount() < 1 or VrGetHypoScore(1) > 60
         or VrGetHypoStr(1) strneq "yes")
         goto retvalue;
            else
repeatagain:
            MediaPlayFile("c:\aaa\prompts\stat_now.wav");


ASMvTop();


#get data


        for (i = 1;i <= nr; i++)
        A[i] = ASGet("pnr_no");
        if (A[i] eq f)
        vid_write("pnr found");
        train_no = ASGet("t_no");
        vid_write(train_no);
        booking_stat = ASGet("booking_status");
        break;
        else
        ASMvNxt();
        endif
        endfor
        Status(booking_stat);
        stringtoDigit(train_no);
```

```
      sleep(10);
       MediaPlayFile("c:\aaa\prompts\repeat.wav");
       VrPlayAndRecogYesNo("C:\aaa\prompts\sayy_n.wav");
       if (VrGetHypoStr(1) streq "yes")
          goto repeatagain;
       else
          goto doyou;
       endif
    endif
doyou:
    MediaPlayFile ("C:\aaa\prompts\continue.WAV");
    VrPlayAndRecogYesNo("C:\aaa\prompts\sayy_n.wav");
    if (VrGetHypoStr(1) streq "yes")
        goto retvalue;
    endif
    VrEndSession();
    endif
  endif
    TrunkDisconnect();
    endprogram
 onhangup
        MediaPlayFile ("C:\aaa\prompts\thanku.WAV");
        TrunkDisconnect();
 end

 func stringtoDigit(digit)
        lenStr = length(digit);
       MediaSetLanguage("English_India");
       MediaPlayFile("C:\aaa\prompts\and.wav");
       MediaPlayFile("C:\aaa\prompts\your.wav");
       vid_write("Your train no is " );
      for(i =1; i <= lenStr; i++)
       digit1 = substr(digit,i,1);
       vid_write(digit1);
              digit1 = digit1 &".wav";
              MediaPlayFile("C:\aaa\prompts\"&digit1);
        endfor
return digit;
endfunc

func pnrvalue(digit)
        lenStr = length(digit);
      MediaPlayFile("C:\aaa\prompts\isur_pnr.wav");
      vid_write("your pnr is " );
      for(i =1; i <= lenStr; i++)
       digit1 = substr(digit,i,1);
       vid_write(digit1);
              digit1 = digit1 &".wav";
              MediaPlayFile("C:\aaa\prompts\"&digit1);
        endfor
return digit;
endfunc

func Status(stat)
vid_write(stat);
        st1 = ctoi(stat);
        st2 = ctoi ("confirmed");
        st3 = ctoi ("RAC");
        st4 = ctoi ("wait_list");
        if (st1 streq st2)
               MediaPlayFile("c:\aaa\prompts\confirm.wav");
               MediaPlayFile("c:\aaa\prompts\coachno.wav");
        vid_write("Your coach number is ");
        coachno = ASGet("coach_no1");
        coachstat(coachno);
        coachno = ASGet("coach_no");
        coachnostat(coachno);
        MediaPlayFile("c:\aaa\prompts\berthno.wav");
        vid_write("Your berth number is ");
        berthno = ASGet("seatno");
        berthnostat(berthno);
else
        if (st1 streq st3)
                MediaPlayFile("c:\aaa\prompts\rac.wav");
                MediaPlayFile("c:\aaa\prompts\coachno.wav");
                coachno = ASGet("coach_no1");
                coachstat(coachno);
                coachno = ASGet("coach_no");
                coachnostat(coachno);
                MediaPlayFile("c:\aaa\prompts\seatno.wav");
                vid_write("Your seat number is ");
                berthno = ASGet("seatno");
                berthnostat(berthno);
        else
                if (st1 streq st4)
                        MediaPlayFile("c:\aaa\prompts\waitlist.wav");
                        berthno = ASGet("seatno");
                        vid_write("Your position is ");
                        berthnostat(berthno);
                endif
        endif
endif
return stat;
```

```
endfunc

func coachstat(coach)
        lenStr = length(coach);
        for(i =1; i<=lenStr; i++)
        coachno = substr(coach,i,1);
        vid_write(coachno);
        coachno = coachno &".wav";
                MediaPlayFile("C:\aaa\prompts\"&coachno);
        endfor
        return coach;
endfunc

func coachnostat(coach)
        lenStr = length(coach);
        if (lenStr eq 1)
                for(i =1; i<=lenStr; i++)
                        coach1 = substr(coach,i,1);
                        vid_write(coach1);
                endfor
                coach1 = coach1 &".wav";
                MediaPlayFile("C:\aaa\prompts\"&coach1);
        else
        for(i =1; i<=lenStr; i= i + 2)
                        coach1 = substr(coach,i,2);
                        vid_write(coach1);
                endfor
                coach1=coach1 &".wav";
                MediaPlayFile ("C:\aaa\prompts\" &coach1);
        endif
return coach;
```

```
endfunc
func berthnostat(berth)
        lenStr = length(berth);
        if (lenStr eq 1)
                for(i =1; i<=lenStr; i++)
                        berth1= substr(berth,i,1);
                        vid_write(berth1);
                endfor
                berth1 = berth1 &".wav";
                MediaPlayFile ("C:\aaa\prompts\" &berth1);
        else
                for(i =1; i<=lenStr; i = i +2)
                        berth1 = substr(berth,i,2);
                        vid_write(berth1);
                endfor
                berth1 = berth1 &".wav";
                MediaPlayFile ("C:\aaa\prompts\" &berth1);
        endif
return berth;
endfunc
```