

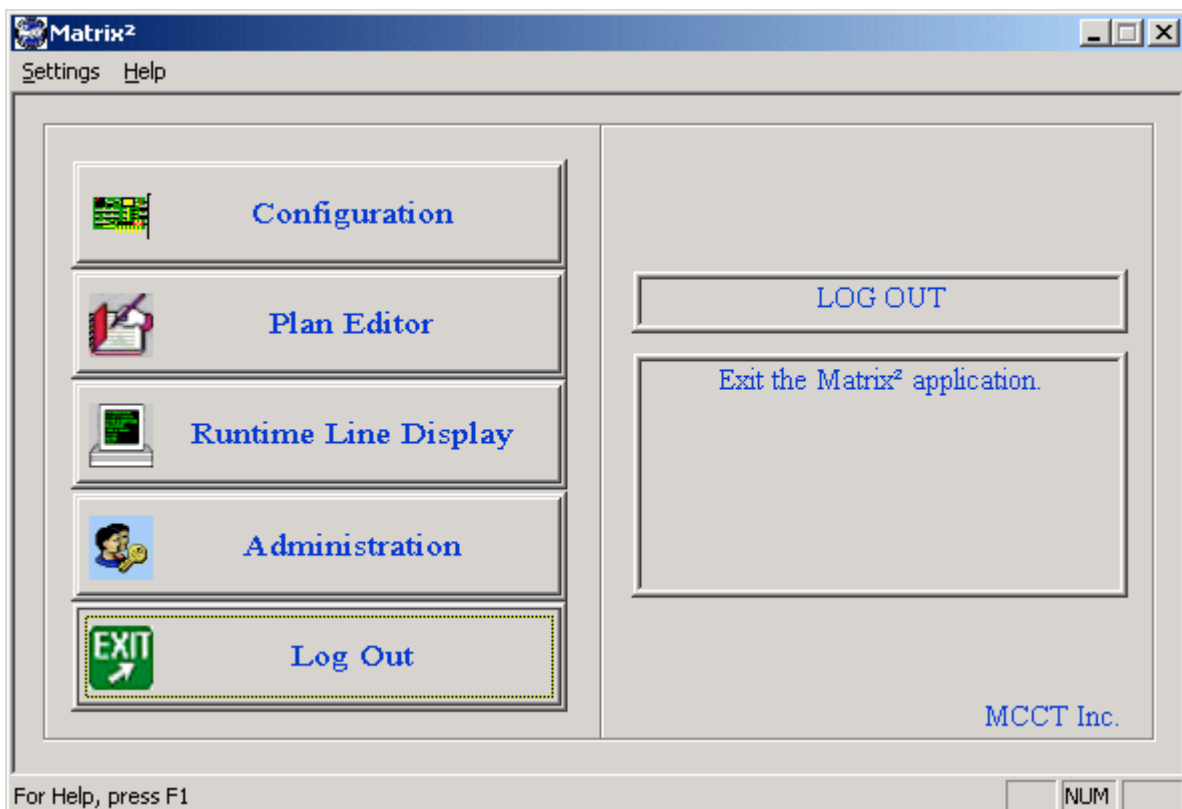
# MCCT, Inc. Matrix<sup>2</sup> Overview

## **I. Introduction**

MCCT's Matrix<sup>2</sup> IVR Platform is an all-inclusive Application Generator system with the expendability to grow to 480 inbound (or outbound) ports, a GUI Windows Plan Editor, built-in Text-to-Speech prompt conversion and much more that will be shown. The Matrix<sup>2</sup> platform uses RedHat Linux as the operating system of choice and applications (Plans) are structured on the robust MySQL database.

Matrix<sup>2</sup> is feature rich with the ability convert text to speech (TTS) for voice prompts & other messages; Continuous Word Speech Recognition; Conferencing with GUI interfaces for administrators & operators; Fax broadcast & Fax-on-Demand and Unified Messaging applications.

The Main Menu for the Matrix GUI Interface is shown below in Figure 1. There are four major categories from which to choose: *Configuration*, *Plan Editor*, *Runtime Line Display* and *Administration*.



**Figure 1. Matrix<sup>2</sup> GUI Interface Main Menu.**

Each of these menu selections will be discussed in the course of this overview.

## Configuration

Four options are in the Configuration menu: *Port Configuration*, *Make Call Block*, *CPA Table* and *DNIS*. Since Dialogic voice boards are no longer being shipped with the Matrix IVR's (Aculab is MCCT's voice board of choice), the *Make Call Block* and *CPA Table* options are no longer required. They remain in the screen so that we can support systems that have shipped with Dialogic boards. Figure 2 shows the initial Configuration Menu screen.

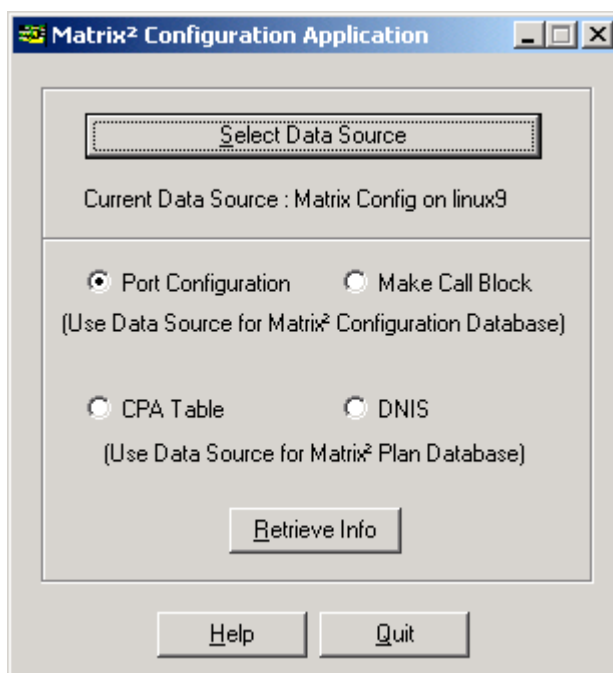


Figure 2. Configuration Menu Screen

Details of the Port Configuration are given in the Matrix<sup>2</sup> User Manual Chapter 2, Section 2-2. Figure 3 shows the Port Configuration screen as well as the Port Edit screen. When Matrix was initially designed, MCCT was using both Dialogic and Aculab voice boards. Support for Dialogic diminished to the point of non-existent after Intel bought them out and we stopped using Dialogic in new systems. although we still support fielded units. The reason for mentioning this is that some of the fields in the Port Configuration screen are Dialogic only. These are: *Board ID*, *Country Protocol* and *Make Call Block*. For Aculab, country protocol is transparent to Matrix and *isdn* is the only entry needed, in lieu of *5ess*, *ni2*, *ctr4* and other specific protocols.

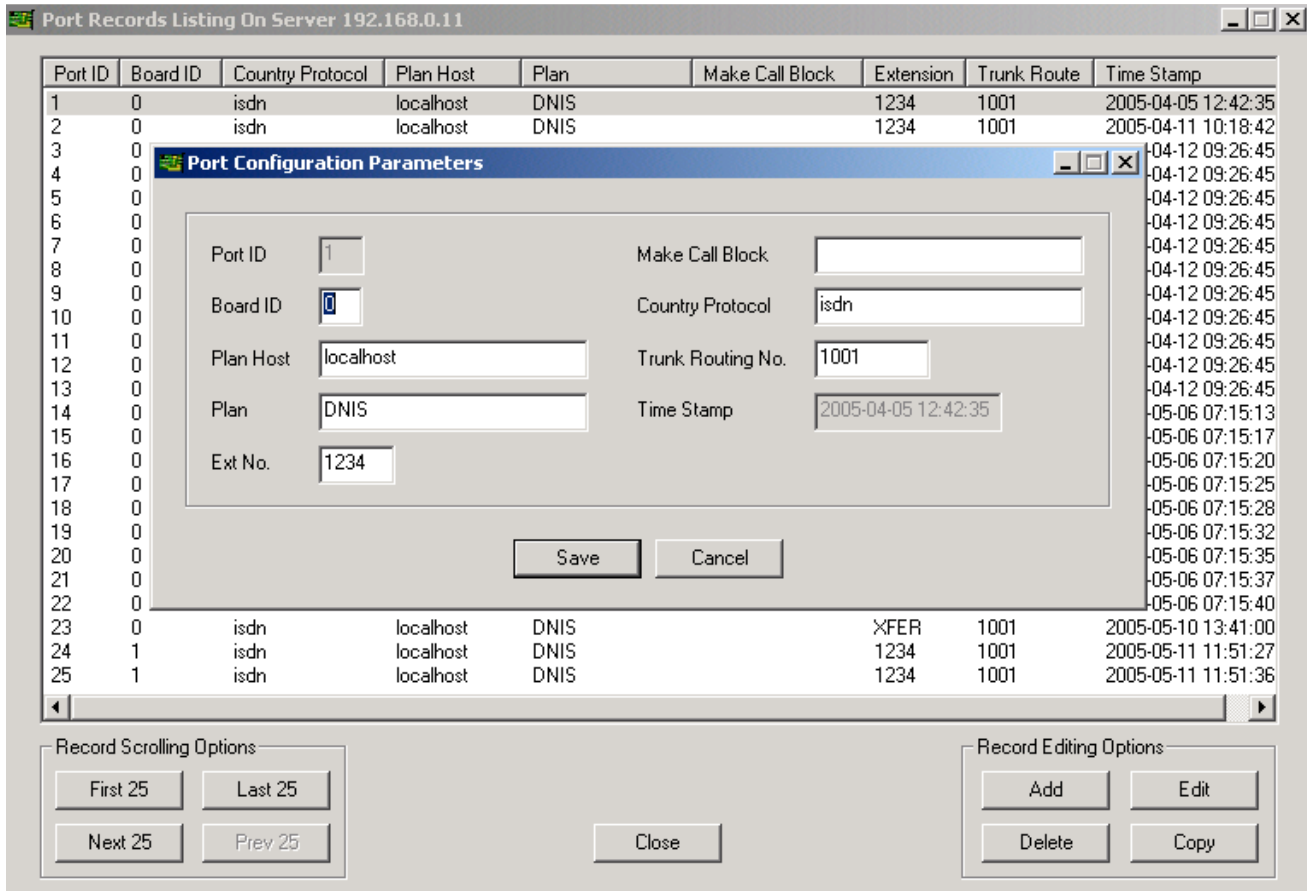
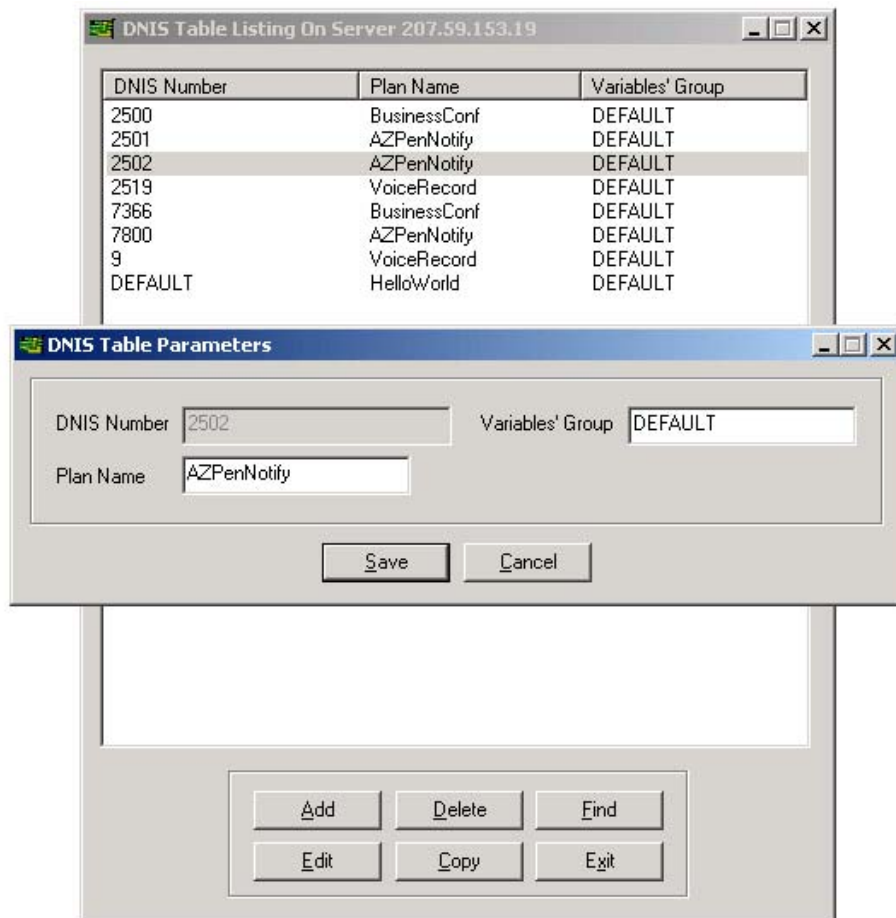


Figure 3. Port Configuration Screen.

In the edit screen, you can change any of the valid fields except for the grayed out *Port ID* field which is the port selected to edit. In the *Plan* field most ports will be set to *DNIS* where the dialed number is "looked up" in the *DNIS Table* to find which Plan is associated with the DNIS. Instead of putting *DNIS* in the Plan field, a Plan name may be entered where the Plan runs on the line.

Since the *Make Call Block* and the *CPA Table* options do not apply to the new IVR's with Aculab Voice boards, these will not be discussed. Details on these Dialogic requirements are contained in the User Manual, Chapter 2, sections 2-3 and 2-4, respectively.

The *DNIS* option in the *Configuration* menu brings up the DNIS table for mapping a specific DNIS number to a Plan. As will be shown in Figure 4, it is a straightforward process to add or edit any DNIS number to the desired Plan. This is the table that is called up if there is a *DNIS* entered in the *Plan* column in the Port Configuration screen; please refer to Figure 3.



**Figure 4. DNIS Table Screen with Edit Screen.**

The *DNIS Numbers* shown in Figure 4 are 4-digit since that is what the CO or Switch is presenting. There will be situations where the number of digits presented is 7, 10 or something else. In every situation, the *DNIS Number* column must reflect the number of digits sent by the Telco or PBX Switch.

### Plan Editor

The windows GUI interface to the MySQL database is in the form of the Plan Editor. Each Matrix<sup>2</sup> command is translated into a 30-column table with the first four columns titled *Step*, *Label*, *Command* and *Display*. The remaining column names are *Parameter 1* through *Parameter 26*; these parameter columns map to the various field in the command screens. Some sample commands will be shown to demonstrate the GUI interface appearance of these screens. You may see all of the commands that currently exist in Revision 2.2 of the Matrix<sup>2</sup> User Manual. This will not be re-stated in each command group paragraph to avoid repetition.

The Plan Editor screen is shown in Figure 5 with the command groups in the left hand screen and the Plan commands in the right hand screen. The commands in the command groups will be shown and highlights of the Plan Editor screen will be discussed. For further details, please reference the Matrix<sup>2</sup> User Manual.

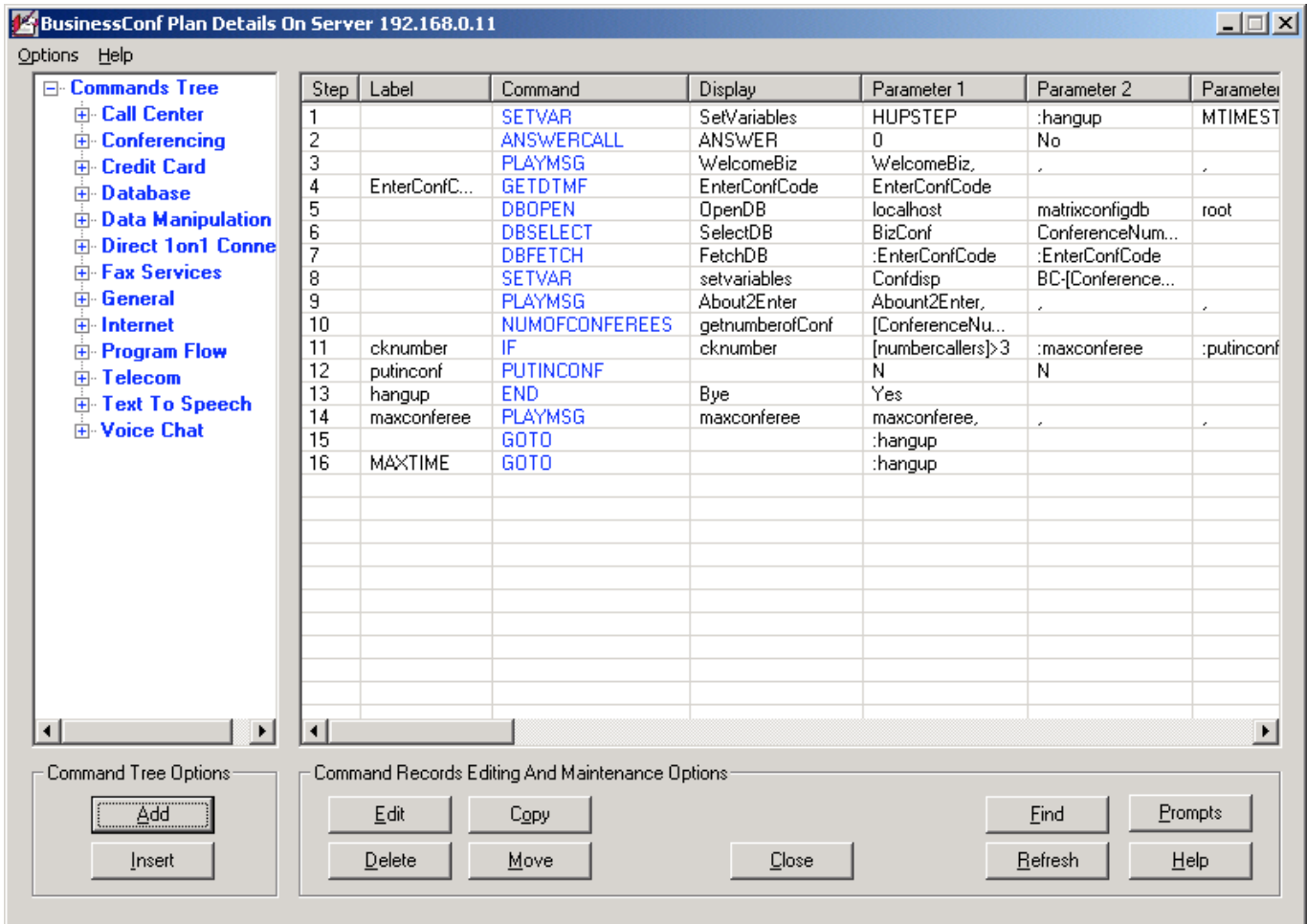


Figure 5. Plan Editor Screen.

In the following paragraphs, each command group is outlined with its associated commands and brief descriptions. In some cases, a command screen will be shown to better communicate the structure and functionality of commands in general.

**Call Center Commands**

While there are commands in this Command Group, they are pre-release and there are more commands planned. At this time, Call Center goals are in development and in the next revision of the User Manual it is anticipated that this Command Group will be complete.

The basic commands shown in the Plan Editor See User Manual include Agent Login/Logout, Agent Authorization and Distribution, Call Queuing (Distribution) and Outbound Connection and Closing. These are building blocks for the remaining commands to be developed.

Queuing is a major feature of the Call Center group, however, it must be tied to Skills Based Routing. Commands that perform this function will be using priority schemes that are configurable by an administrator. Other Call Center command functions will include finding agents based on personal preference, skills or idle times and grouping commands that put agents in groups based on skills.

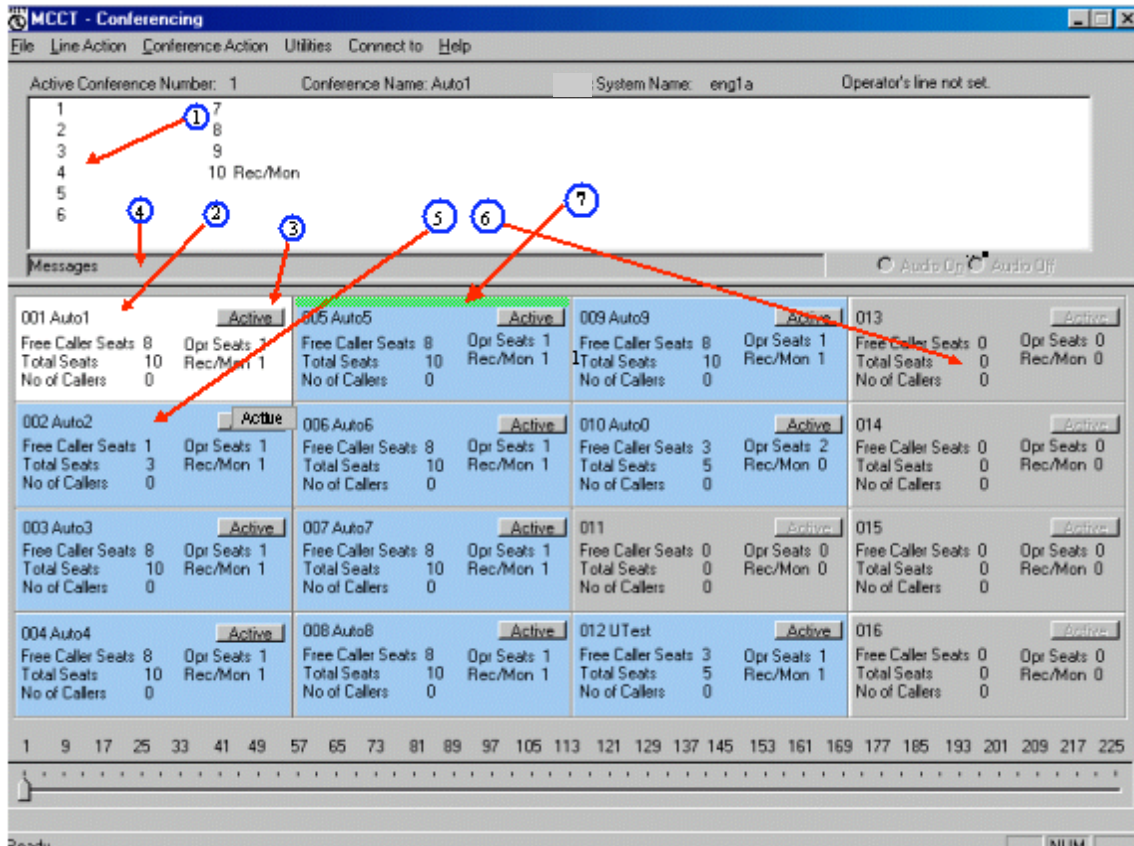
The commands shown in the Plan Editor are placed there for testing purposes and at the present time do not represent a complete set. If you have any questions or comments, please contact MCCT at 1-800-GO-4-MCCT (Domestic U.S.), 603 524-2214 (International) or [jd@mcct.com](mailto:jd@mcct.com) (Director of Engineering).

### ***Conferencing Commands***

The *Conferencing* command group allows you to set up and configure 2-party to 64-party conferences in any way you like. The conferencing commands are: *EAVESDROPCONF*, *GETFREECONF*, *NAMECONF*, *NUMOFCONFEREES*, *PLAYINCONF*, *PLAYTONECONF*, *PUTINCONF* and *SC1ON1*. These commands provide the means of placing callers in a conference (pre-determined or chosen), find a free conference, allow two callers to go into a private 1-on-1 conference, and much more. A monitor may also listen in to a conference to see if it is still active, for example. This leads to another conferencing feature available from MCCT.

As an added feature for conference administration, MCCT has developed a GUI windows interface for administrators, monitors and/or operators to manage all of the possible conferences. As many as 240 conferences may be configured and conference size can range from 2 to 64 people in a conference. Seats within a conference may be reserved for the number of callers, monitors or operators. The monitor or operator (or an administrator) may function as quality control for the conference. Callers have the option to hit the "0" key to ask for help; the operator will be able to pull that caller into a private room (to avoid disruption of the on-going conference) and discuss the callers issue. As an example, a caller may want to ask a question regarding conference protocol or about when the conference topic will be scheduled again.

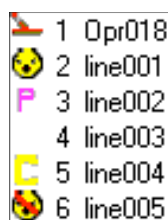
The means by which the operator moves the caller and herself to a private room is simply with a mouse drag and drop. This is one of the many features of the MCCT Conferencing GUI Interface. Other features include setting callers as *Pupils* or as a *Coach*. *Pupils* can hear the *Coach*, but not each other; only the *Coach* can hear the *Pupils*. The operator may also mute one or all of the callers. Figure 6 shows the Conferencing GUI Interface screen.



**Figure 6. Conferencing GUI Interface Screen.**

Each conference window contains six fields and an *active* bar. The *active* bar is pointed to with the mouse and clicked to enter that conference window. In each window are: a) the conference number and associated name; b) the number of free caller seats; c) the total seats; d) the total number of callers (this includes operators) e) the total of Operator seats and f) the recording/monitoring device. As seen on the previous page, the conferencing screen has seven numbered arrows that point to items of note. These items are described below. It is important to know that within a conference, the numbers in the top screen are *seat* numbers - not line numbers; the line numbers are shown after the seat number when there are active callers.

1. This section of the MCCT Conferencing Screen is where the callers with their appropriate Line Number are seen. Operators (or monitors) show the monitor belt icon and instead of showing 'Line nnn' show 'Opr nnn' where 'nnn' is the three-digit line number in both cases. Normal callers will have a yellow 'face' icon when they are active – or talking - followed by the Line Number. Callers set up as Pupils will have the purple 'P' as an icon while a caller set up as a Coach will have the yellow 'C'. In the figure below, Seat Number 4 has a caller who is not designated as Pupil or Coach and is not talking. The Caller in Seat Number 6 has been muted.



2. The white background in the Conference 001 window represents this window as the *active* conference and thus the activity there is shown in the top portion of the screen.
3. This is the *Active* button which, when clicked, brings the selected Conference up with a white background and shows the status of all assigned seats.
4. This is the message bar and contains information that is meaningful to the operator. An example of a message here could be: "*Help has been requested in Conference 1 for Seat 2.*" If the operator had been in conference 003, this message would be an alert to enter conference 001 for assisting seat #2. If requested, time in conference is also shown here.
5. Since the white background corresponds to the conference that is active, blue backgrounds are for the inactive conferences (Inactive does not mean that there is nothing happening in the conference; it means that there is no activity shown in the top portion of the screen). When an inactive conference is full, there is a red background.
6. Gray backgrounds indicate that the conference has not been set up; note that there are no values given for any of the fields in the window.
7. When there are callers in the conference, the Record/Monitoring "seat" - or device - can be activated by clicking on the "Start Recording" selection in the Conference Action drop-down menu (If there are no callers, these menu selections will be grayed out). When recording is active, there will be the green bar shown in Figure 6, Conference 005.

### ***Credit Card Commands***

At this time, there is one Credit Card command – *CCMOD10*. This command is basically a screening command that executes the MOD10 formulas to determine if the credit card is valid or not. Future credit card commands will include fields for merchant account data, credit card type(s) & numbers and links to the credit card processing sites (or telephone number fields to process over a modem).

## ***Database Commands***

This command group communicates directly to a selected MySQL database and requires some knowledge of MySQL syntax. The seven *Database* commands are:

- a) DBCLOSE
- b) DBDELETE
- c) DBFETCH
- d) DBINSERT
- e) DBOPEN
- f) DBSELECT
- g) DBUPDATE

Based on these commands, the actions that can be taken in a Plan include functions such as opening & closing a database; inserting, deleting & getting table data and updating table data. Most applications use databases extensively and these commands facilitate the communication with the appropriate database. There are three basic databases used in Matrix:

- a) *matrixconfigdb* – contains all configurable tables including user accounts, conferencing setup, port configuration, System and User Variables and so on.
- b) *matrixplandb* - this database contains all of the Plans and associated commands and the Call Detail Record (CDR) tables.
- c) "*Custom DB*" - in most Plans the customer's application requires a special database to manage the specifics of the Plan. As an example, an application where there are several customer accounts would require tables for addresses, PIN's, phone numbers, purchased items or services, and so on in this Plan specific database.

## ***Data Manipulation Commands***

There are two commands in this group: *SETLINEVAR* and *SETVAR*. These are used to set variable names and their corresponding values (initial). The first of the two is for special situations where a Plan is running on a specified line instead of DNIS and a DNIS Table lookup.

Variables are one of the key ingredients in a plan. They are like "data buckets" that store data of all types based on caller input or system information. As an example, if a caller enters a DTMF value for a PIN, that PIN number would be stored in some variable name and be associated with the time and line number of the call.

### ***Direct 1on1 Connect Commands***

This is another specialty command group with two commands: *DCONNECTSTART* and *DCONNECTION1*. These commands are used to establish 1on1 connections to callers who will be placed in a two-seat conference. There are a variety of uses for this type of application such as help desks, etc.

### ***Fax Services Commands***

This new command group contains the two commands needed for fax receipt and transmission: *FAXRECEIVE* and *FAXSEND*. Please reference the white paper titled "Aculab\_Fax\_White" for more details on the MCCT Matrix<sup>2</sup> fax architecture using "tif" files and being able to convert from all other types of files. Using other Matrix<sup>2</sup> commands, Plans can be setup to broadcast faxes, have fax-on-demand or receive faxes from callers.

### ***General Commands***

This is one of the larger command groups containing commands that perform a variety of functions including playing & recording a message, creating comments to help others understand the flow of the Plan and a command for the hangup with the option to write a call detail record (CDR) for the call. The commands in this group (Chapter 9 of the User Manual) include: *COMMENT*, *END*, *ENDSPAWNPLAN*, *HOOK*, *LINUXCMD*, *LOG2FILE*, *PLAYMSG*, *RECORDMSG*, *SPAWNPLAN* and *WAITMSEC*.

To better understand the GUI appearance of Matrix<sup>2</sup> commands, the *PLAYMSG* command screen is shown on the next page in Figure 7. There is the possibility for ten messages to be played in sequence in this one command. As an example, each successive message can consist of a string of related subjects such as a general introduction, followed by a sponsor name, followed by an ad, followed by a phone number and so on. The messages can be variables where each one is rotated for every call or determined by the DNIS or ANI of the call.

Other features of this command include the ability to determine the format of the message. The prompts may be in a character, currency, date, literal, number, text-to-speech or time format. These are described in more detail in chapter 9 of the Matrix<sup>2</sup> User Manual. The messages may also be set to be interruptible so that the caller may press a key to stop the message and go on to the next. An example of this is a long introductory message that plays every time may be stopped by the caller so that s/he may hear the next messages more quickly. A last feature is that a tone may be played with the message in the event that the application requires some enhancement to go with the prompt.

No.	Prompt Name	Format	Interruptible?	Play Tone?
1.	[EnterANI]	Character	No	No
2.			No	No
3.	CallingFrom		No	No
4.			No	No
5.	[City]	Literal	No	No
6.			No	No
7.	County		No	No
8.			No	No
9.	[County]	Literal	No	No
10.			No	No

**Figure 7. Sample *PLAYMSG* Command Screen.**

This sample command from the General Command Group represents a typical command screen with the basic attributes:

- Display and Label fields;
- Fields to fill in a value, variable or expression;
- Drop-down fields to choose an option;
- Previous, Save, Cancel and Next buttons.

Other sample commands will be shown to show a larger complement of attributes. As you will see, the user-friendliness of the Plan Editor GUI interface makes writing an application very straightforward.

### ***Internet Commands***

This new group at present has the one command that created the need for this group: *INTERNETSSL*. This command provides the means for connecting to a site via Secure Sockets Layer (SSL); these connections are encrypted for protection of the data being exchanged. The site(s) are typically validation sites for credit cards or some other transactions where the URL, Port Number, Page Path and Data Stream are provided. This command provides all of the fields needed and as a result streamlines the process. The alternative is writing C-code in a hook or some other code format.

### ***Program Flow Commands***

Commands in this group determine the flow of the application based on the Plan author's setup within the Plan, specific conditions being met (or not) or the choices made by the caller. There are five commands in this group: *GOTO*, *IF*, *JUMPBACK*, *JUMPOUT* and *SELECT*. The *GOTO*, *JUMPBACK* and *JUMPOUT* are "within-the-Plan" flow commands where the *IF* command tests for "if/then/else" conditions and the *SELECT* command provides choices to the caller. The *SELECT* command is shown in Figure 8.

In the *SELECT* command screen, notice that there are twelve fields for prompts that can be played sequentially, just as was described in the *PLAYMSG* command in the *General* command group. There are also twelve fields for choices available to the caller (0-9, #, \*). The choices are given in the prompt(s) and there may be from two to twelve selections defined.

There is also a field in this command for Speech Recognition. Should the caller be given the option to "Speak or Say . . .", the choices are defined in this field as the "grammar" words/characters that are possible. This would include 0-9, #, \* and other possible words such as "operator", "agent", "call", "reserve" and so on. The remaining fields are the standard entry fields, drop-down boxes and buttons seen on most of the Matrix<sup>2</sup> commands.

Program Flow SELECT Command

Display  Label

Parameters

Prompt Names

To Be Played

Store DTMF In  Prompt On Invalid Key

No. Of Retries  Interruptible Message?

On Maximum Retries Go To Plan  To Label

Key	Go To Plan	Go To Label	Key	Go To Plan	Go To Label	Key	Go To Plan	Go To Label
0	<input type="text"/>	<input type="text"/>	4	<input type="text"/>	<input type="text" value="[Opt4]"/>	8	<input type="text"/>	<input type="text" value="[Opt8]"/>
1	<input type="text"/>	<input type="text" value="[Opt1]"/>	5	<input type="text"/>	<input type="text" value="[Opt5]"/>	9	<input type="text"/>	<input type="text" value="MaintPW"/>
2	<input type="text"/>	<input type="text" value="[Opt2]"/>	6	<input type="text"/>	<input type="text" value="[Opt6]"/>	*	<input type="text"/>	<input type="text"/>
3	<input type="text"/>	<input type="text" value="[Opt3]"/>	7	<input type="text"/>	<input type="text" value="[Opt7]"/>	#	<input type="text"/>	<input type="text" value="MainMenu"/>

Voice Recognition

No. Of Timeouts  Waiting Prompt During Timeout

Timeout Interval  Timeout Go To Plan  To Label

Disconnect Voice Resource?

Figure 8. The **SELECT** Command Screen.

**Telecom Commands**

These commands are about the handling of the actual telephone connection. The functions include making the connection, dialing out, getting DTMF tones and disconnecting the call. The seven commands in this group are: *ANSWERCALL*, *CONNECT*, *DIAL*, *GETDTMF*, *OUTDISCONNECT*, *QUERYTRUNK* and *ROUTE*. Each serves its own unique function; they described in detail in the Matrix<sup>2</sup> User Manual, Chapter 11. An example of one of these commands, *DIAL*, is shown in Figure 9.

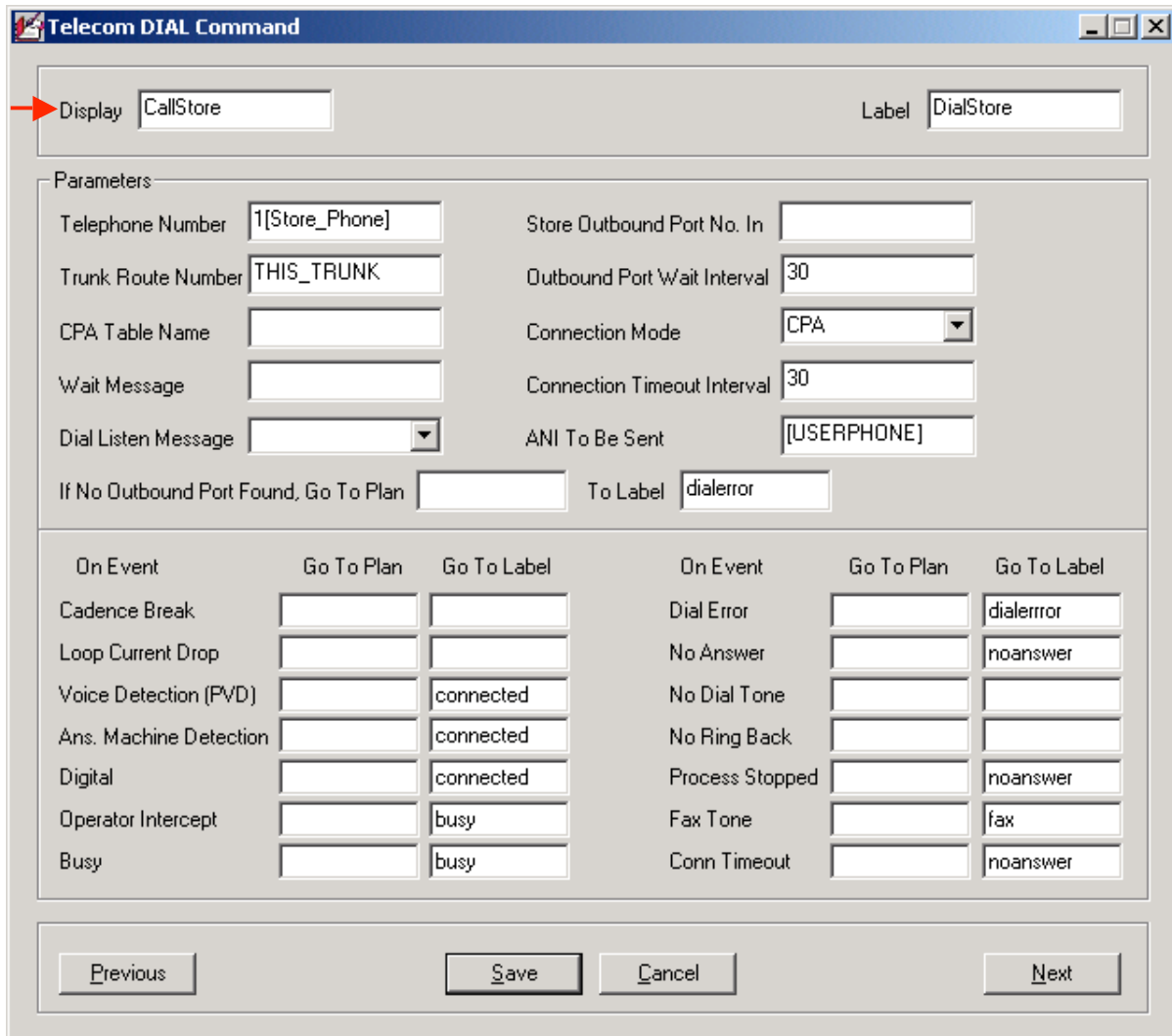


Figure 9. The **DIAL** Command Screen.

Notice that in this command screen, there are several “Event” conditions that can determine the flow of the Plan. In this one, the *Connection Mode* is CPA which is Call Progress Analysis. This sets up all of the detection schemes for situations such as Answering Machine, Fax Tone, No Answer, Busy, etc. These are all configurable using the CPA parameters in the Aculab setup. When any one of these is detected, the Plan flow will go to the Label that is called out for that detection mode.

**Text To Speech Commands**

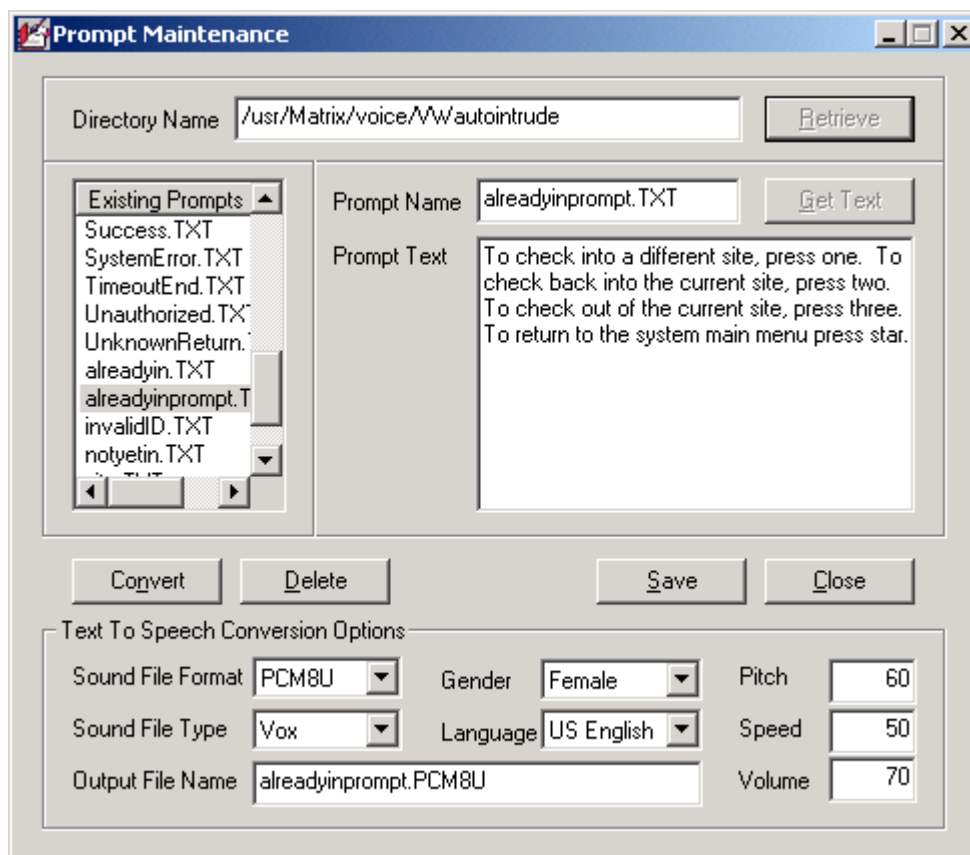
The commands here: *FILETOFILECONV*, *SETSOUNDBANK*, *SETSPEECHPARAM*, and *VARTOFILECONV* are intended for limited use in the flow of a Plan – primarily for test and development. While these commands function in the same manner as the manual method using the *Prompts* button (at the bottom of the Plan Editor screen), there are normally two TTS channels shipped with an IVR thus only two active callers could “convert on the fly”. Additionally, conversion in the flow of a Plan takes time and taxes the flow. More discussion follows in the “Prompts Button” section.

## Voice Chat Commands

There are eighteen commands in this special command group devoted to Voice Chat applications. Please reference the Matrix<sup>2</sup> User Manual, Chapter 13, for more details as the subject is too extensive to be included in this Overview.

## Prompts Button Utility

The *Prompts* button brings up the screen shown below in Figure 10. Here, a prompt name can be created, the text of the prompt typed in, the conversion options chosen and the conversion can then be made by clicking on the *Convert* button.



**Figure 10. Prompts Screen for TTS conversion to a File.**

In the *Text To Speech Conversion Options* portion of the screen, the Sound File Format and Type can be selected. Additionally, the gender (Any, Female, Male) can be chosen and the language. You may also vary the Pitch, Speed and Volume of the voice by changing the percentage in each field (default percentage is 50).

This is a very valuable tool for developers and Plan authors. Without requiring the need for a special application to call in and record prompts (or record them off line and transfer them to the IVR), simply typing them and converting them is all that is needed.

Runtime Line Display

All active calls and their progress may be observed using the *Runtime Line Display*. On one screen you may view fifty calls at a time and view the next- or previous-50 by clicking on the *Next* or *Previous* button. Up to 480 calls may be active on a system at one time. The *Wait Call* state in the Display column indicates that the line is idle and ready to receive a call. In the *Plan Name* column, there is either, *DNIS* meaning that the next call will do a look up in the DNIS Table to match the DNIS number to a Plan, or there is the Plan Name on an active call where the table lookup matched the Plan Name shown (e.g., *BulletinBoard*). In the *Display* column for an active call, the *Display* field of a command (see the red arrow in Figure 9, "The *DIAL* Command Screen") is shown. Figure 11 shows the *Runtime Line Display* screen.

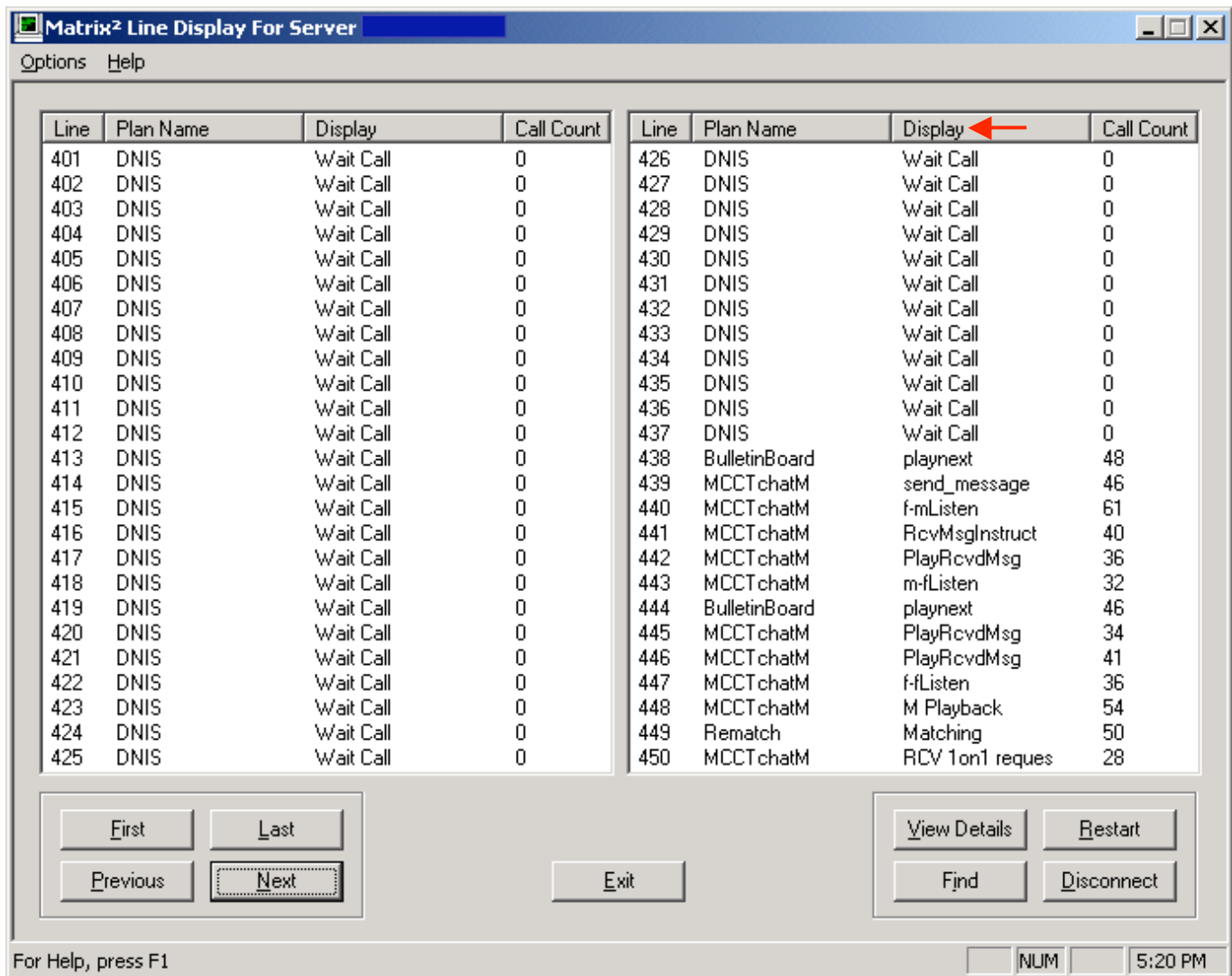


Figure 11. *Runtime Line Display* Screen.

Another feature of the *Runtime Line Display* is the “View Details” button. On any highlighted line, clicking this button will show all of the specific data for the caller on that line. Information such as time called, DNIS, ANI, Plan Name and much more can be retrieved using this feature. Figure 12 shows a sample of the data that can be gotten. In this example, the values for the ANSWER\_TIME, DNIS and LINE\_NO User Variables are shown.

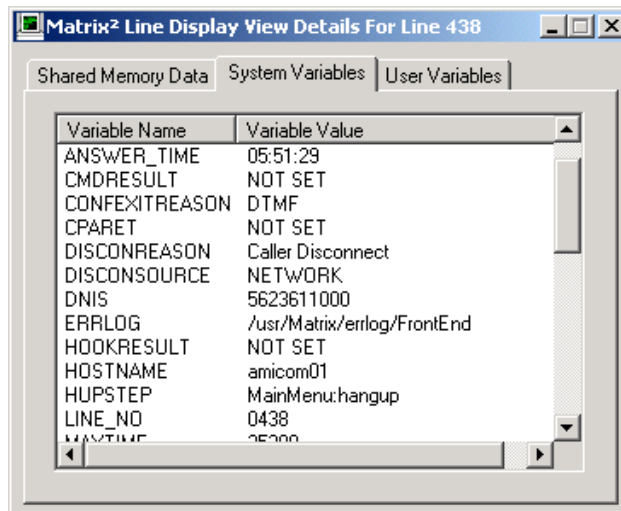


Figure 12. View Details Information.

Administration

The fourth major area of the Matrix<sup>2</sup> Main Menu is *Administration*. There are three “tabs” in the *Administration* screen: User, Plan and System. The User tab shows the user privileges granted for the four Matrix<sup>2</sup> categories: *Admin*, *PortConfig*, *Plan Editor* and *LineDisplay*; Figure 13 shows the privileges granted to each user.

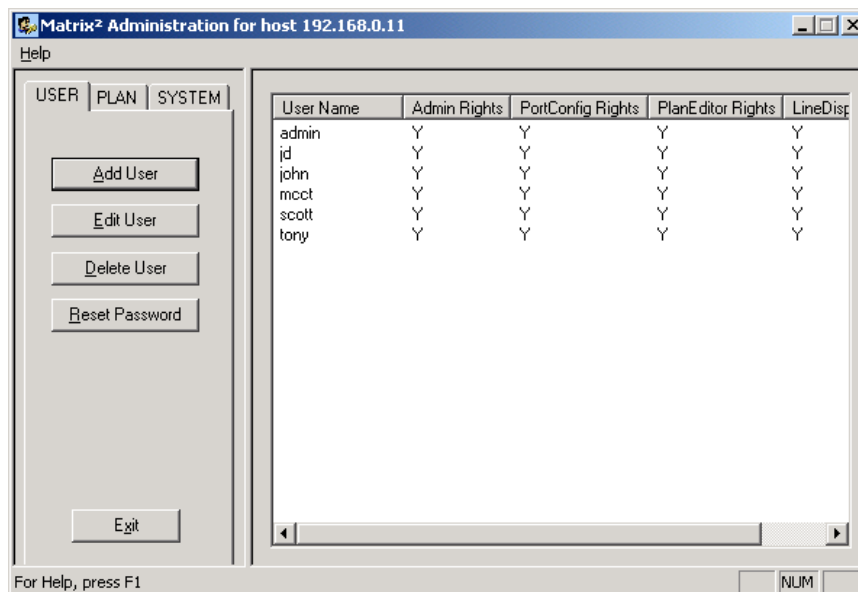


Figure 13. Administration – User Screen.

The *Plan* screen allows the user (with Admin privileges) to copy a part or all of one Plan to another. Examples of usage include copying an existing Plan to another Plan Name that is similar to make minor changes for a different result. You may, for example, copy the Plan "sports\_football" to "sports\_baseball" and edit the copied plan to tailor it for baseball. You must first create the empty Plan "sports\_baseball" before copying "sports\_football" to it. This is done with the *Add Plan* button in the *Administration – Plan* screen. Each Plan will reside in a selected DSN (Data Source Name); this may be on the same IVR or two different IVR's. The *Administration – Plan* screen is shown in Figure 14.

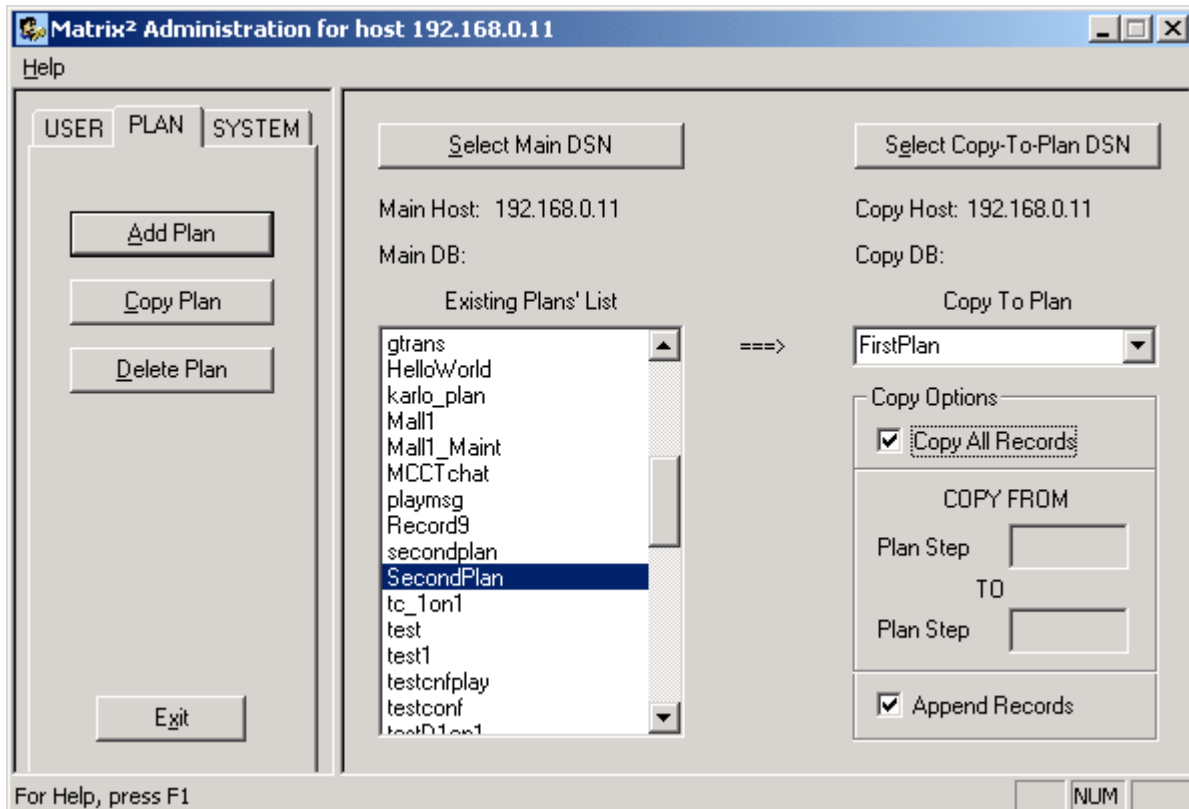
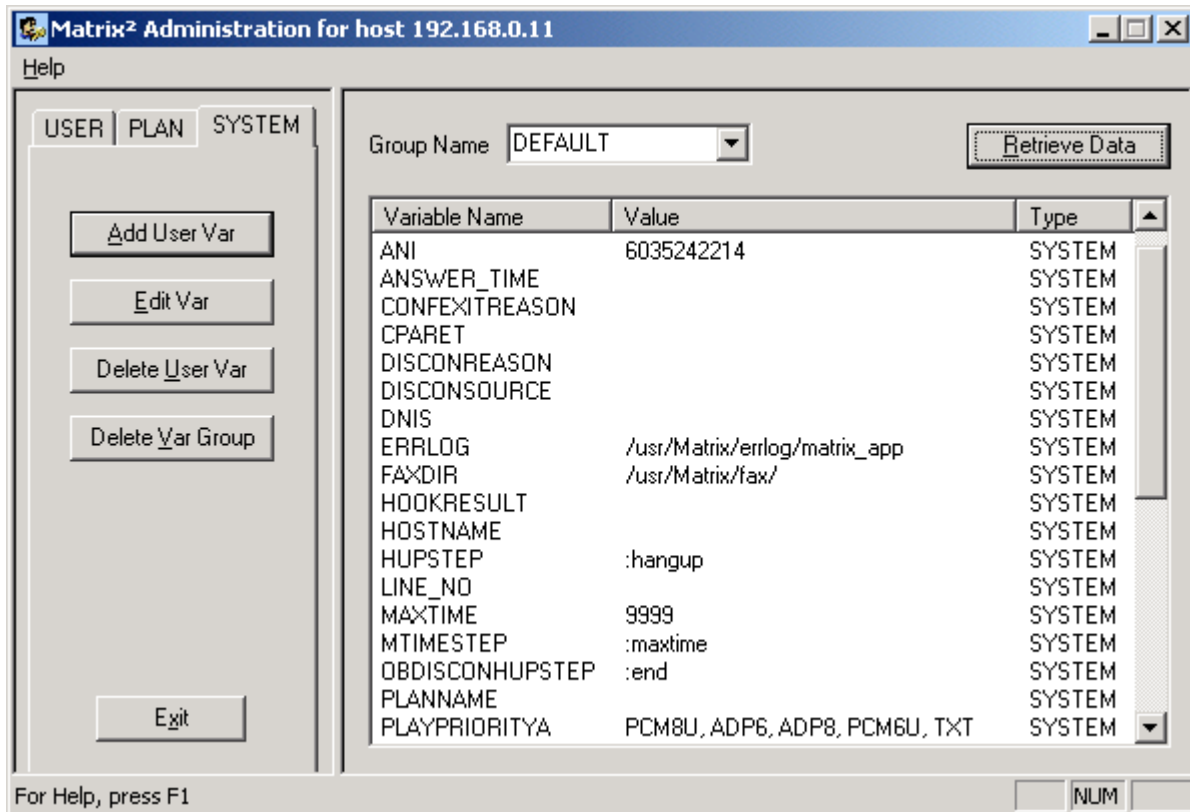


Figure 14. *Administration – Plan* Screen.

The last part of the *Administration* screen is the *System* tab. Here are listed all of the System and User Variables. The *Default* Variable Group includes all of the standard System Variables that are a part of each Plan; customized Groups may be added for specific Plans or groups of Plans. When customization occurs, the added variables are User Variables, defined by the Plan author. Additional variables may be added within a Plan that are not shown in the Variable Group. This is accomplished by use of the *SETVAR* command in the *Data Manipulation* Command Group.

Figure 15 shows the *Administration – System* screen.



**Figure 15. Administration – System Screen.**

## II. White Papers

In this section, a series of white papers is presented on the following topics:

- A. Matrix<sup>2</sup>/Aculab Continuous Word Speech Recognition,
- B. Matrix<sup>2</sup>/Aculab Fax,
- C. Matrix<sup>2</sup>/Aculab Conferencing and
- D. Them versus Us.

## A. Matrix<sup>2</sup> Speech Recognition

### I. Introduction

MCCT introduces the addition of Aculab's Connected-Word Speech Recognition (CWR) to the already robust Matrix<sup>2</sup> product. This value-added feature allows callers to respond to application prompts using either the existing DTMF method or the new Automatic Speech Recognition (ASR) method. This ASR feature broadens selection possibilities and adds a new dimension of interaction between the caller and the IVR. Application programmers will now be able to create Plans that are more comprehensive with less complicated approaches.

This paper presents the ASR feature as it is used in Plan Editor commands with examples of how it can be implemented. Familiarity with the Matrix<sup>2</sup> Plan Editor and its commands will enhance the reader's understanding of this paper. Please contact MCCT for a copy of the Matrix<sup>2</sup> User Manual.

### II. Aculab ASR Overview

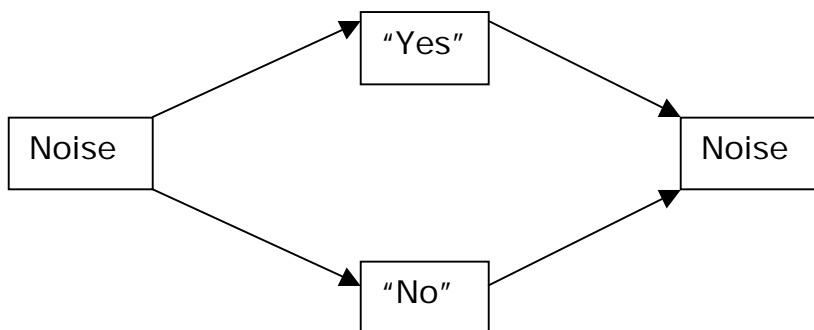
The main elements of Aculab's Automatic Speech Recognition are a phoneme-based recognizer, Connected-Word Speech Recognition server (*ASRserver*) and the 'Grammars'. The terms used will be described in the course of this overview.

A phoneme is a discrete unit of sound in a given language. For example, the word *fish* contains three phonemes written with the Aculab phoneme symbols *f*, *ih* and *Sh* (triphones). An Utterance is a sequence of sounds produced by the caller that consists of phoneme groupings (words) delimited by pauses. Connected-word recognizers process each utterance as a discrete entity. In the case of a caller being prompted to say a 6-digit PIN, each number in the PIN must be separated by a pause that is long enough to be recognized as a delimiter.

The Grammar defining the words to be recognized and their order is specified using Aculab speech grammar format (ASGF) notation – a subset of Java speech grammar format. A discussion of Aculab's speech recognition structure follows.

A *Network*, loaded from a file or built from ASGF, contains a set of nodes and defines the connections between them. Each node refers to a speech recognition model and allows the likelihoods calculated from each model to be accumulated over time. The nodes and their connections are generated from a grammar which is a text string describing the vocabulary which the recognizer must listen for.

As an example, a Network capable of recognizing "yes" and "no" could use a model set containing the two whole word models "yes" and "no", together with an internal noise model to allow for the possible pauses before and after each word.



Exactly two networks are required for a recognition task. One of these defines what may be recognized as spoken data and is termed the "Vocabulary Network". An example of this is the network shown above. The other defines what is recognized as silence and is termed the "Silence Network". A recognition result cannot be generated until the Vocabulary Network outperforms the Silence network. The Silence Network is fixed and must be present in the "net" sub-directory of the ASRserver's working directory.

A Network may be constructed by either using the standalone *ASRNetMan* Windows application (discussed later in this section) or at run time using the appropriate ASR API calls. Each of these methods requires the same inputs (described in the next paragraphs):

- One or more Lexicon and/or Rule objects
- A Model Set object
- An Aculab Speech Grammar Format (ASGF) string

A *Lexicon*, loaded from a file, contains a set of words, each of which is mapped onto one or more alternative sequences of models. These model sequences are specified via sequences of phoneme symbols. There can be more than one pronunciation for each word. Examples are below:

- read:        riyd / rehd
- live:        lihv / liyv

A Lexicon may be constructed or modified using the standalone *ASRLexMan* Windows application (discussed later in this section) which can also display a list of the allowed phoneme symbols for each language.

When creating new lexicon entries, it is generally best to use *ASRLexMan* to look up the pronunciation of a similar-sounding word or words and use a copy-and-paste approach to make up the pronunciation of the new word.

A *Rule*, loaded from a file, contains the rules required to map any word onto a sequence of phonemes (termed “pronunciation by rule”). This method is more comprehensive than any lexicon; it will generate a reasonable pronunciation for any word.

Of course, given the nature of human language it will often make mistakes. That is why rules should only be used as a fallback, for words that are not in the lexicon and are not known at the time the application is designed. It is better to manually extend or customize the lexicon than to rely on the rules since few languages follow an exact set of rules – exceptions are very common.

A *Triphone Map*, loaded from a file (e.g. eng-us-full.rtf) contains a mapping between phonemes and their triphone models, required when building Networks that are to be recognized using an enhanced model file (e.g. eng-us-enhanced.rmf). The triphone mapping will be described further in the discussion on *ASRLexMan*.

### Windows User Interfaces

The two major windows applications are *ASRLexMan* and *ASRNetMan*. These may be downloaded from [www.aculab.com](http://www.aculab.com) in the Software Downloads section under Support on the top menu bar (you must register first with a user name and password). Scroll down the page until you come to Automatic Speech Recognition (ASR) and click on that. On the ASR page, click onto the *Windows* folder; download the .zip file and the two .txt files in a new folder created for ASR. Here you can unzip the file that creates nine folders. The executables for each of the two applications are located in folders of the same name (*ASRLexMan* and *ASRNetMan*).

Each application will be discussed for its usage and functionality; double clicking on the .exe files will bring up the blank screens. To use them, you will need to transfer files from the Linux ASR platform to your PC to load into the programs. These files can be found in the /usr/aculab/cwsw/ASRServer directory under three subdirectories: *lex*, *model* and *rul*. This directory structure is shown below:

```

/usr/aculab/cwsw/ASRServer
[root@linux1 ASRServer]# ls
ASRServer          eng-us-full.rtf      model              rul
asr_server.cfg     lex                  net                SessionManager.o
ASRService.o      main.o              nohup.out         Session.o
ConfigFile.o      makefile.ASRServer  orig               svi

```

Files needed for loading the applications are as follows:

**lex:** eng-us-full.rtf (ASRLexMan – Lexicon/ASRNetMan –Pronunciation Files - Load)

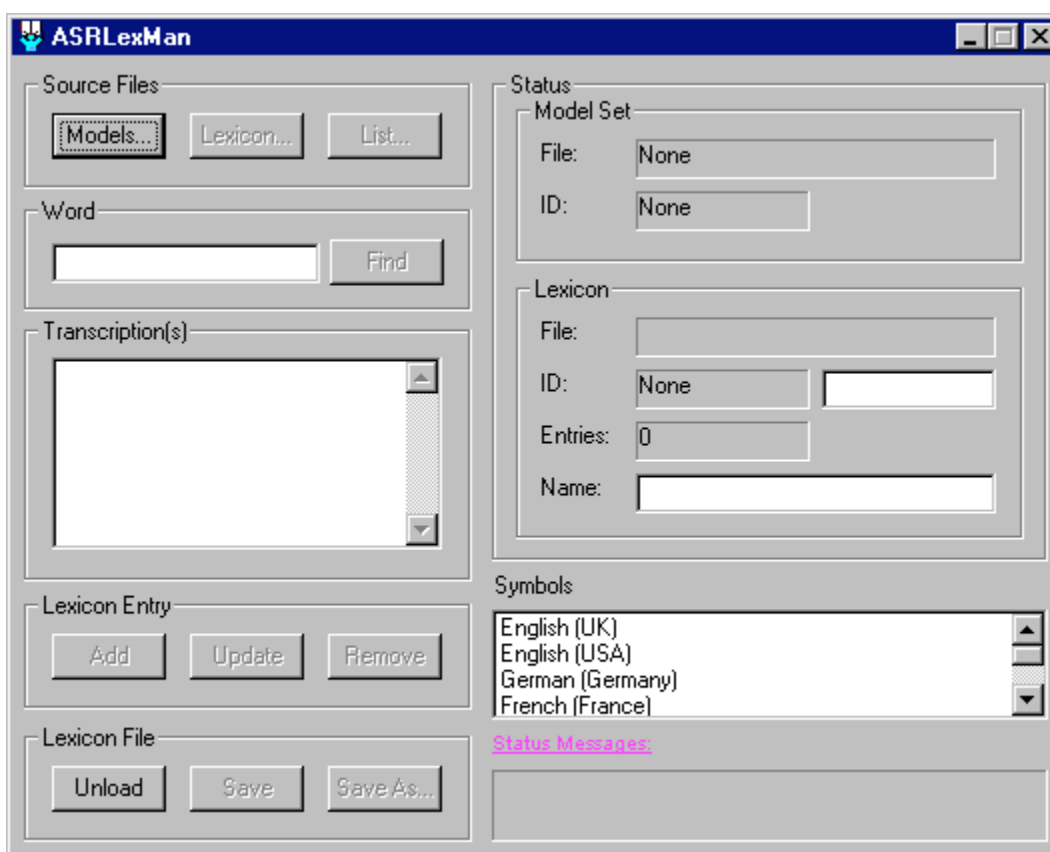
**model:** eng-us-basic.rmf (ASRLexMan – Source Files – Models)

**rul:** eng-us-full.rtf (ASRNetMan – Pronunciation Files – Load)

In the ASRNetMan screen, there is a section in the lower left called “ASGF” with a load button. You can create a text file with notepad and rename it as a .asgf file; this is for the purpose of creating a grammar text string for words to use in your application. An example of a grammar text string (or .asgf file) is shown below:

```
(Sussenberger/macomber/dean/john/operator/yes/no/help/one/two/three/four/five/six/seven/eight/nine/zero/star/pound/a/b/c);
```

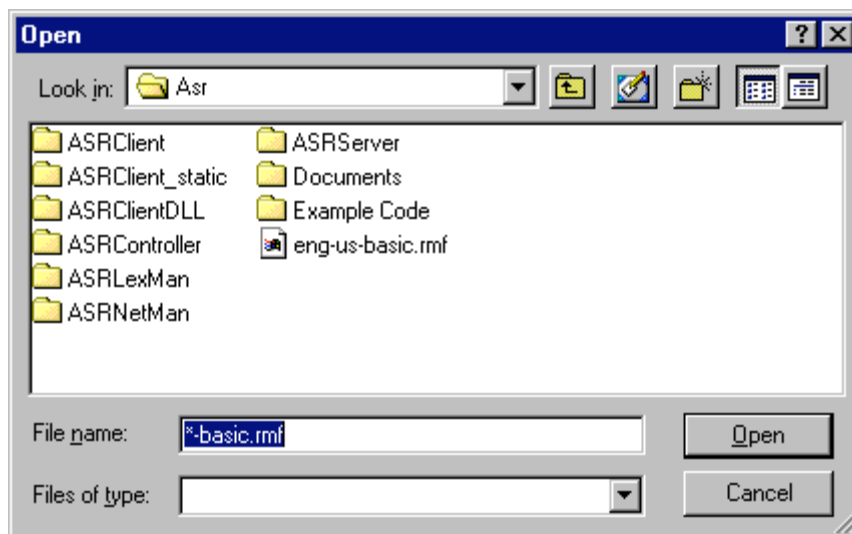
Placing all of these files discussed above in your main ASR directory makes it convenient to load them using these applications. Each application knows the file types to load, so when a “Load” or “Models” button is clicked, only valid files are shown. The ASRLexMan screen is shown in Figure A-1.



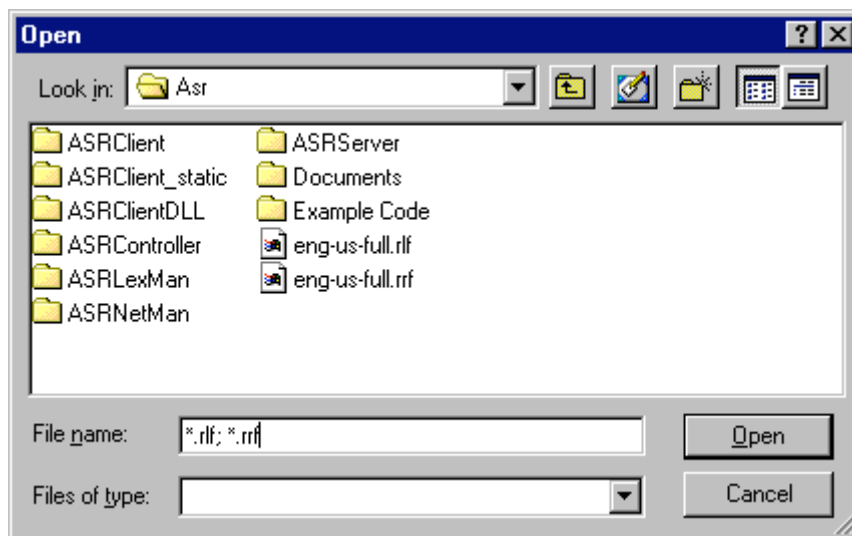
**Figure A-1. ASRLexMan Screen.**

In the ASRLexMan application, the model file must be loaded first; this will make the Lexicon button active. Clicking on the “Models” button shows the valid file to load in Figure A-2; all other files that were discussed in the three sub-directories on the previous page are in the same directory.

**Note:** The file types follow a certain pattern where the middle letter in the three digit extension is key. The .rlf files are *lexicon* files; the .rmf files are *model* files and the .rrf files are *rules* files.



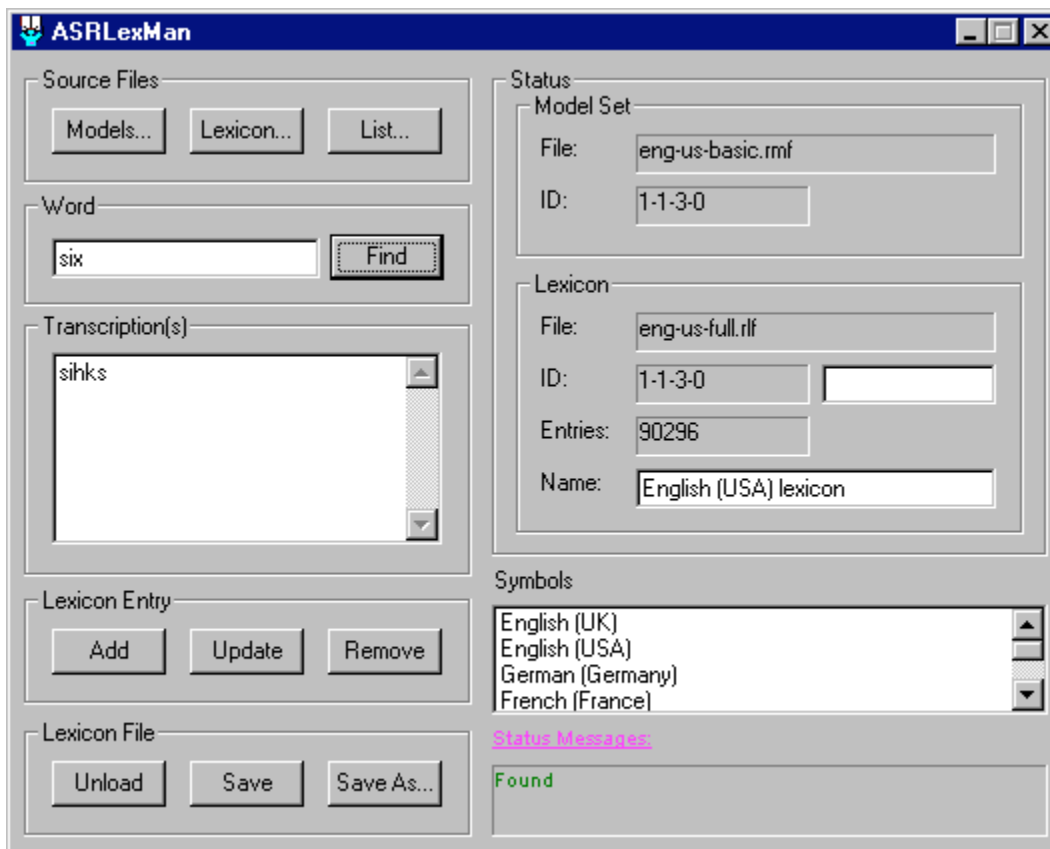
**Figure A-2. Valid File for “Models” Button.**



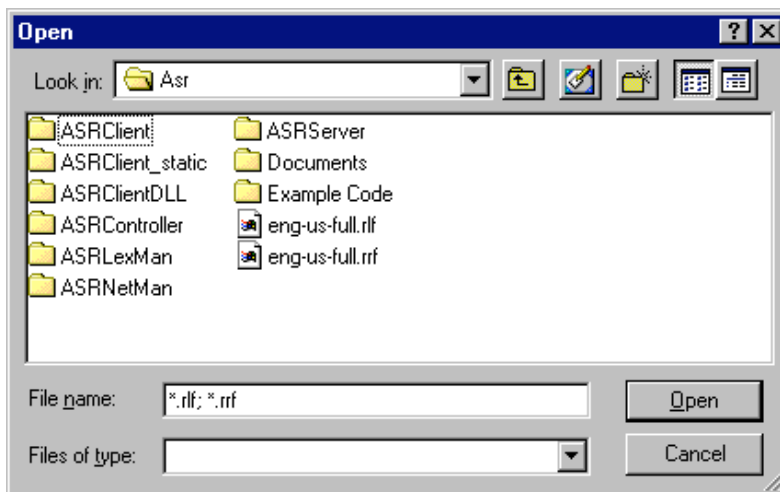
**Figure A-3. Valid Files for Lexicon Button.**

In Figure A-3, there are two file types shown: .rlf (Lexicon) and .rrf (Rules). The best approach is to use the .rlf file as this loads the standard ASR phonemes. You may, however, load the .rrf file should you wish to construct your own pronunciation to be recognized. The .rrf file is also used to construct words not contained in the Lexicon.

Figure A-4 shows the ASRLexMan screen after the Models and Lexicon buttons are loaded. A search for the word “six” shows the Lexicon pronunciation phoneme that was found. Note: When loading the Lexicon button, it will take a few seconds before the screen loading is complete since there are over 90,000 word entries.



**Figure A-4. ASRLexMan Screen Loaded.**



**Figure A-5. Valid Files for ASRNetMan.**

As with the ASRLexMan – Lexicon loading, the valid files for ASRNetMan are .rlf and .rtf as shown in Figure A-5. Depending on whether or not you wish to modify or add pronunciation to a word in the Lexicon or use the Rules file to add a word pronunciation to the Lexicon, you will load one of these files at a time.

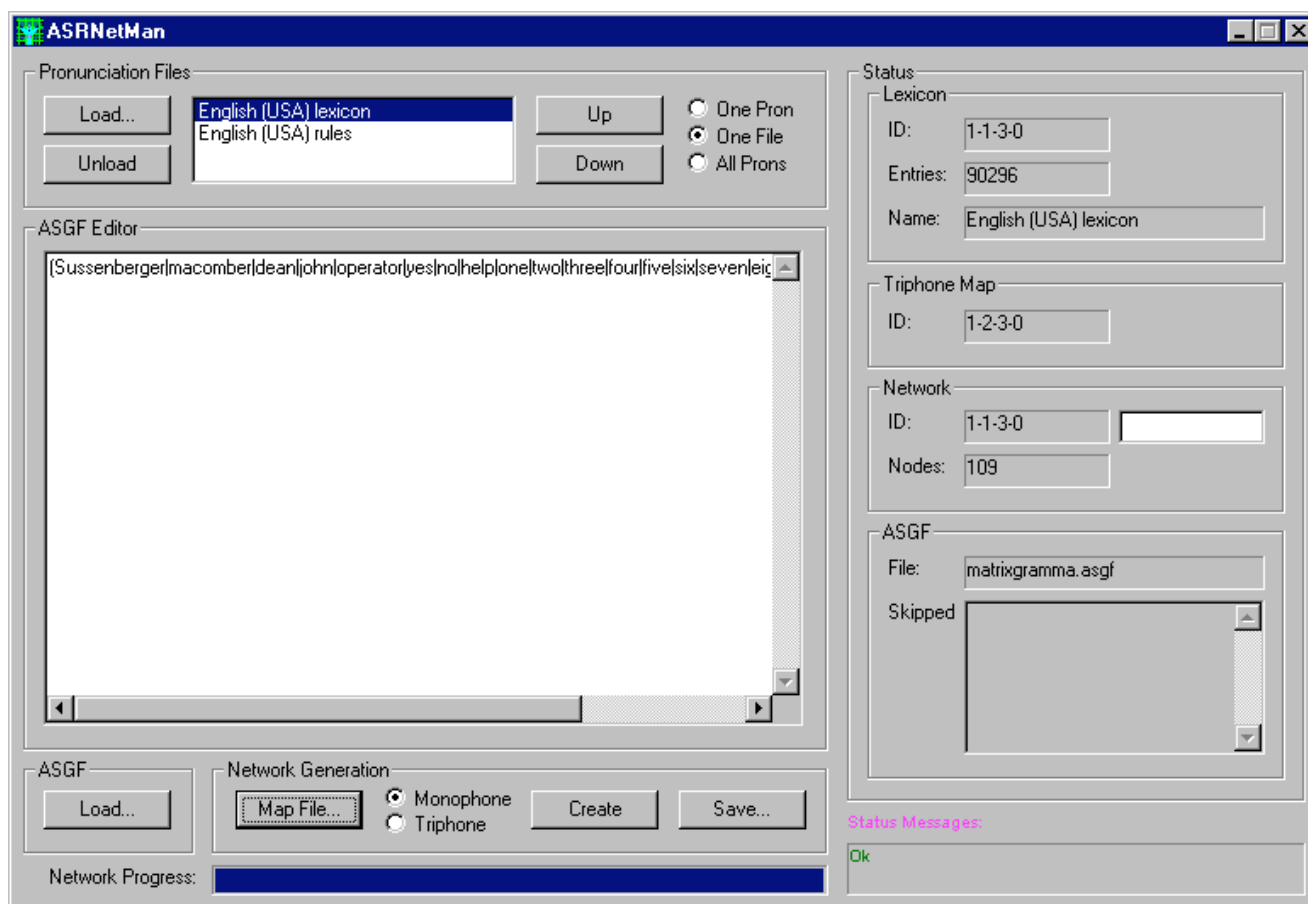


Figure A-6. ASRNetMan Screen – Loaded.

## Application Usage

### ASRLexMan

This application provides the means of managing CWR Lexica. In this context, a lexicon is a list of words, each with a transcription associated with it. A transcription is a list of automatic speech recognition (ASR) models to be used to recognize the word. These models can be a sequence of phonemes (a pronunciation) or one or more whole-word models, or any mixture of the two.

The transcriptions should be entered as a single concatenated string, without any separator. For instance, in US English, the transcription for “hello” is “hehlow”, representing the phoneme sequence “h”, “eh”, “l”, “ow”.

When performing ASR, the incoming audio is processed and then parsed by a recognition network (such as may be produced by ASRNetMan). This Network will have been generated from a grammar definition in ASGF such as “yes | no”, while a lexicon such as *eng-us-full.rlf* will have provided the phonetic transcription of each word in the grammar. In this way, the network contains knowledge of how each word may be pronounced. There can be more than one way of pronouncing a word and all likely possibilities should be included in the Lexicon.

Although Lexica are the preferred method for generating recognition networks, word pronunciation can also be predicted using the Rules provided in files such as *eng-us-full.rrf*. These Rules can be loaded into ASRLexMan in place of the Lexicon. This can give the user an “educated guess” as to the pronunciation of words not explicitly listed in the Lexicon.

After Loading the Models and Lexicon files, ASRLexMan is ready for modifying any lexicon in the file. To look up a transcription for a given word, type that word in the *Word* field and select *Word – Find*. If the word is present in the Lexicon, its transcription(s) will be displayed in the *Transcription* field; if not, the field will remain empty. If a Rule file has been loaded, rather than a Lexicon, the *Transcription* field will display the predicted pronunciation, which the user can then correct (if necessary) and add to a Lexicon by going back to loading the Lexicon file.

To modify the transcription for a given word, or add an alternative pronunciation, first look it up and then modify the contents in the field. While typing, keep an eye on the *Status Messages* to see whether or not your transcription consists of valid symbols. If more than one transcription is required, each should be entered on a new line, i.e., separated by a <CR>. Once the transcription is correct, the entry may be updated by selecting *Lexicon Entry – Update*. Use this same technique in the *Transcription* field if a word lookup fails and you wish to add a pronunciation to a new word in the Lexicon.

To add a list of new words and their transcriptions in an efficient manner, the information may be entered into a list file. Create a text (.txt) file using Notepad and enter all pertinent information. This can be loaded by selecting *Source Files – List*. In this case, when a word is already in the Lexicon, the supplied transcription will be added to the lists of transcriptions associated with that word, unless it is a duplicate in which case it will not be added.

There are many words that are mispronounced in society and provisions can be made in ASRLexMan to allow for this. The word “nuclear” is correctly pronounced as “new klee ar” whereas many say “new kuw lar”. Figures 7 and 8 show how you would add the incorrect pronunciation so that it will be recognized since it is mis-spoken by a multitude of people – including high ranking officials. Figure 7 shows adding the new transcription and the “invalid transcription” message in the *Status Messages* box. By continuing the new transcription to add the last “r”, the message changes back to “valid transcription”. Notice that with an “invalid transcription” message, the *Add* and *Update* buttons are grayed out. By adding the last “r”, the message changes to “valid transcription” and the two buttons become active.

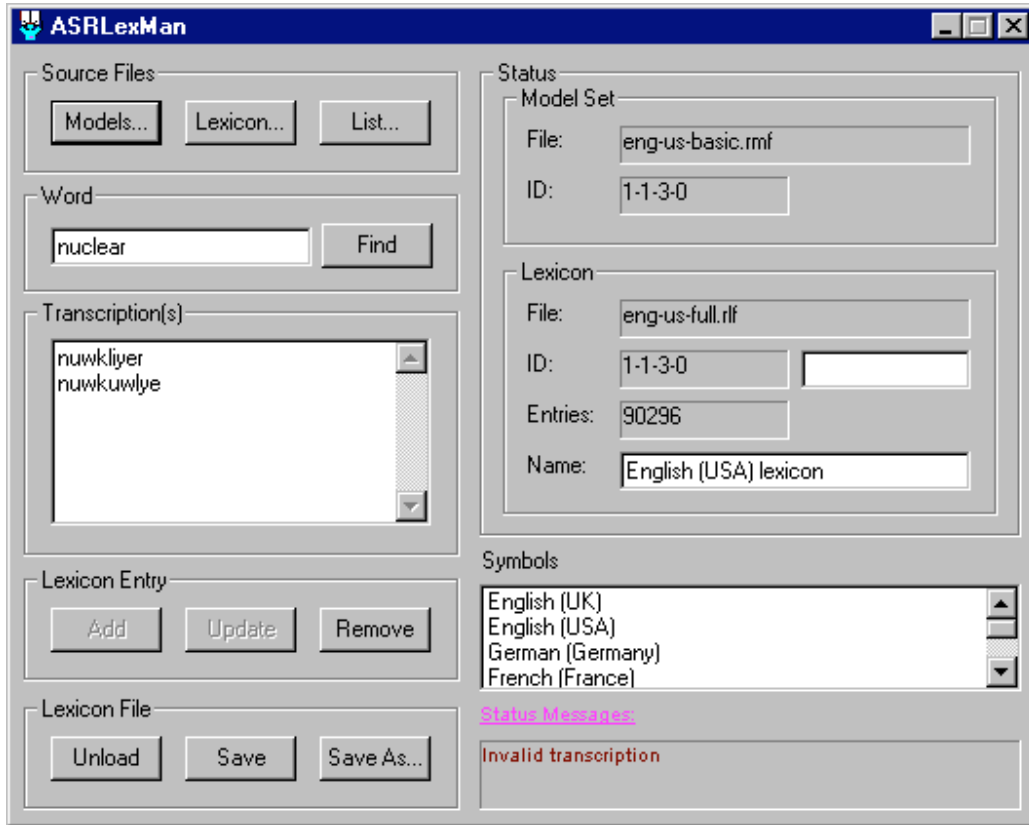


Figure A-7. Adding Transcription – Invalid

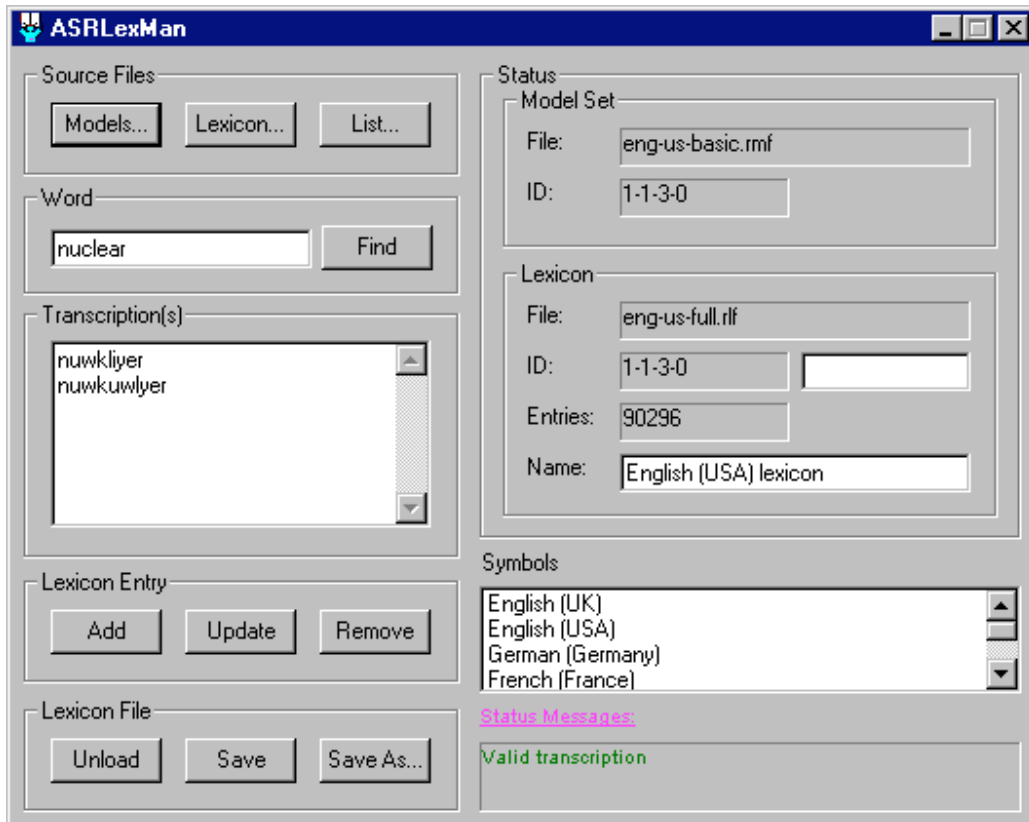


Figure A-8. Adding Transcription – Valid

## ASRNetMan

The ASRNetMan application allows a user to manage Connected Word Speech Recognition (CWR) networks. A Network is a structure against which processed incoming audio is parsed and recognition results obtained. It is generated from an ASGF (Aculab speech grammar format) definition, one or more lexica, an optional Rule file and (usually) a triphone map file. Figure 6 shows the ASRNetMan screen.

If more than one pronunciation file is loaded, the way in which they are searched may be modified. First, the files are searched in the order in which they appear in the *Pronunciation Files* list; in Figure A-6, the files are "English (USA) lexicon" and "English (USA) rules". To move a file up or down in the list, highlight it and click on the "Up" or "Down" button.

If the "All Prons" option is selected, then, for a given word, all transcriptions from all loaded pronunciation files are used as alternative pronunciations. This is only recommended if all the pronunciation files are lexica because the pronunciations predicted by Rule are not always accurate.

If the "One Pron" option is chosen then, for a given word, only the first transcription found *in each file* is used.

If the "One File" option is picked then, for a given word, only the transcription(s) appearing in the first file (in which lookup is successful) is used.

The options "One Pron" and "One File" may be set together so that only a single pronunciation is found.

If a Rule file is included in the list, it is recommended that:

1. It should be last in the list,
2. Only one other pronunciation file should be used, and that should be a Lexicon,
3. "One File" should be set so that the Lexicon is always used in preference to the Rules and
4. One Pron should be cleared so that any multiple entries in the Lexicon are all used, allowing for variation in pronunciation.

As was previously discussed, an ASGF grammar string may be loaded. This can be entered manually in the *ASGF Editor* field or a text file with an extension of .asgf may be loaded using *ASGF – Load*.

Note: You can create a .txt file in Notepad, save it and rename it from filename.txt to filename.asgf.

Building the Network is the next step. If triphone models are to be used at run-time, select *Network Generation – Map File* to load a triphone map file. For US English this should be *eng-us-enhanced.rtf*. Normally, the *Network Generation – Triphone* option should then be selected; otherwise the Network will be built to work with *monophone* models.

If no triphone map file is loaded, a monophone network can still be built, but that should only be considered if there are severe system constraints (triphone models give much higher recognition accuracy).

The Network is built by selecting *Network Generation – Create*. If the Network is large or complicated, look at the *Network Progress* bar to monitor progress and *Status Messages* to see what is happening and if any warnings or errors were generated. If any words are not found in the lookup, they are skipped and, as far as possible, the Network will have been built around them.

Any skipped words will be listed in the *ASGF Skipped* field. However, if any words are listed there they must be treated as an error and either:

- The grammar should be edited to correct any spelling mistake(s),
- The Lexicon should be extended with ASRLexMan to include the skipped word(s) or
- A Rule file should be added to the end of the pronunciation file list to predict the pronunciation of any out-of-lexicon word(s).

Once the Network creation is complete, you must save it using *Network Generation – Save*. Figure A-9 shows a saved triphone Network.

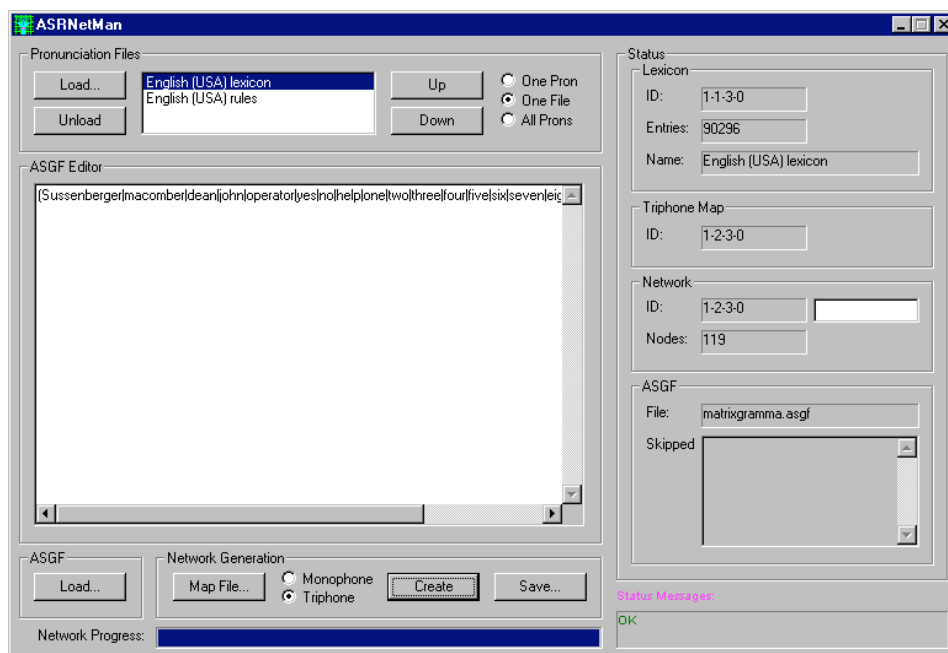


Figure A-9. Saved Triphone Network

### III. Matrix<sup>2</sup> Speech Recognition Implementation

The Matrix<sup>2</sup> Plan Editor contains a multitude of commands for creating a “Plan” or an application. Initially, some of these commands were designed to collect DTMF from the caller for such things as selecting an option, entering a PIN or credit card number and so on. The two most common Matrix<sup>2</sup> commands for collecting DTMF are: *Select* and *GetDTMF*. These command screens are shown in Figures 10 and 11, respectively.

**Program Flow SELECT Command**

Display  Label

Parameters

Prompt Names

To Be Played

Store DTMF In  Prompt On Invalid Key

No. Of Retries  Interruptible Message? No

On Maximum Retries Go To Plan  To Label

Key	Go To Plan	Go To Label	Key	Go To Plan	Go To Label	Key	Go To Plan	Go To Label
0	<input type="text"/>	<input type="text"/>	4	<input type="text"/>	<input type="text"/>	8	<input type="text"/>	<input type="text"/>
1	<input type="text"/>	<input type="text"/>	5	<input type="text"/>	<input type="text"/>	9	<input type="text"/>	<input type="text"/>
2	<input type="text"/>	<input type="text"/>	6	<input type="text"/>	<input type="text"/>	*	<input type="text"/>	<input type="text"/>
3	<input type="text"/>	<input type="text"/>	7	<input type="text"/>	<input type="text"/>	#	<input type="text"/>	<input type="text"/>

Voice Recognition

No. Of Timeouts  Waiting Prompt During Timeout

Timeout Interval  Timeout Go To Plan  To Label

Disconnect Voice Resource? Yes

Figure A-10. *Select* Command Screen.

The screenshot shows a dialog box titled "Telecom GETDTMF Command". It has a standard Windows-style title bar with minimize, maximize, and close buttons. The dialog is organized into several sections:

- Display:** A text input field.
- Label:** A text input field.
- Parameters:** A section containing five small text input fields.
- Store DTMF In:** A text input field.
- No. Of Digits:** A text input field.
- Error Go To Plan:** A text input field.
- To Label:** A text input field.
- No. Of Timeouts:** A text input field.
- Waiting Prompt During Timeout:** A text input field.
- Timeout Interval:** A text input field.
- On Timeout Go To Plan:** A text input field.
- To Label:** A text input field.
- Disconnect Voice Resource?:** A dropdown menu currently set to "Yes".
- Interruptible Message?:** A dropdown menu currently set to "No".
- Terminate DTMF With:** A dropdown menu.

At the bottom of the dialog, there are four buttons: "Previous", "Save", "Cancel", and "Next".

Figure A-11. *GetDTMF* Command Screen.

The *GetDTMF* command is the same as the original; no fields have been added to allow for Speech Recognition values at this time. This is currently handled by another means and will be discussed later in this paper. The *Select* command has had the addition of the line shown between the red arrows in Figure A-10. This new line can contain a string of words separated by a "|" delimiter; these words are what make up the valid "Grammar" words for this Plan or application.

### ***Select* Command**

In the DTMF-only mode, the *Select* command was limited to twelve choices per command: 0 – 9, \* and #. More choices required additional *Select* commands. Using an example of an application where callers can select one of thirty ring tones to be downloaded based on their account information; in the DTMF-only mode three *Select* commands would be required. With Speech Recognition, this can be done in one *Select* command with a single voice prompt. In multiple *Select* commands, three separate voice prompts would be required.

In the next few pages, an example of how the *Select* command can be used with Speech Recognition will be shown. This will involve other Matrix<sup>2</sup> Plan Editor commands to show program flow and functionality. For more details on the Plan Editor commands, please contact MCCT Inc. for a Matrix<sup>2</sup> User Manual.

The example involves callers in a Plan where they can purchase items and have them shipped to their home. Each caller will have an account with billing and ship to information based on their PIN and user name. The items for this example are team pennants for major league baseball teams. In the *Select* command, team names with two words will be chosen in the 0 – 9, \*, # section. All other team names that are single words will be chosen by saying the team name; this will be set up in the new 'Voice Recognition' line.

**Program Flow SELECT Command**

Display  Label

Parameters

Prompt Names

To Be Played

Store DTMF In  Prompt On Invalid Key

No.Of Retries  Interruptible Message?

On Maximum Retries Go To Plan  To Label

Key	Go To Plan	Go To Label	Key	Go To Plan	Go To Label	Key	Go To Plan	Go To Label
0	<input type="text"/>	<input type="text"/>	4	<input type="text" value="White_Sox"/>	<input type="text"/>	8	<input type="text"/>	<input type="text"/>
1	<input type="text"/>	<input type="text" value="Red_Sox"/>	5	<input type="text"/>	<input type="text"/>	9	<input type="text"/>	<input type="text"/>
2	<input type="text"/>	<input type="text" value="Blue_Jays"/>	6	<input type="text"/>	<input type="text"/>	*	<input type="text"/>	<input type="text"/>
3	<input type="text"/>	<input type="text" value="Devil_Rays"/>	7	<input type="text"/>	<input type="text"/>	#	<input type="text"/>	<input type="text"/>

Voice Recognition

No.Of Timeouts  Waiting Prompt During Timeout

Timeout Interval  Timeout Go To Plan  To Label

Disconnect Voice Resource?

**Figure A-12. Example Select Command.**

The voice prompt *Pennant* is shown in Figure A-13 on the following page.

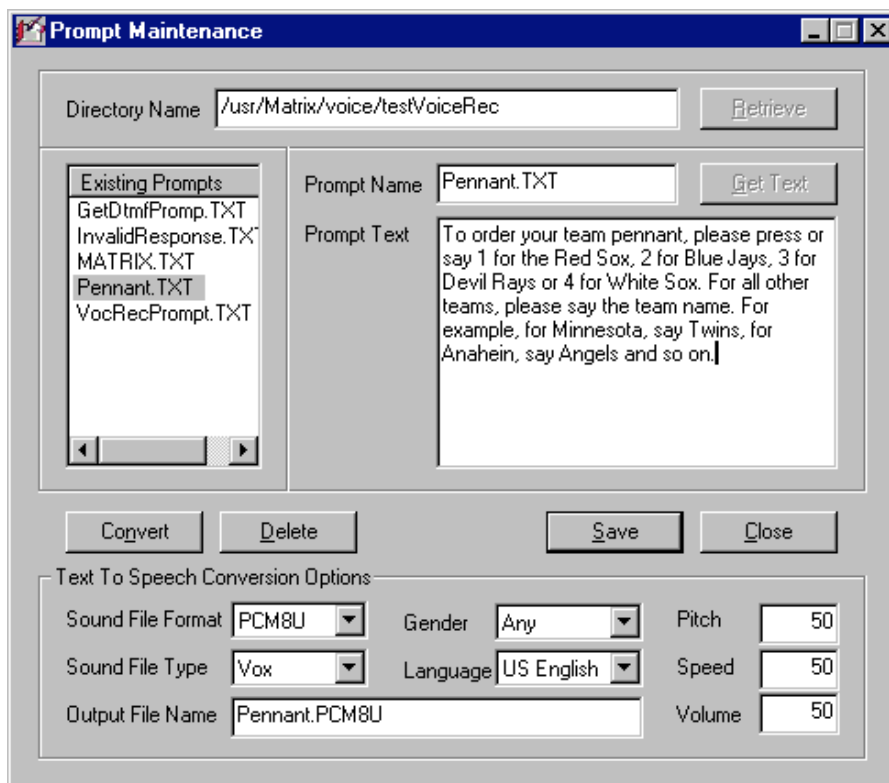


Figure A-13. Voice Prompt for *Pennant?* Select Command.

For teams with single names, the program flow goes to a series of *IF* commands to add the chosen team name to the callers account to bill and ship. Figure A-14 shows the first *IF* command.

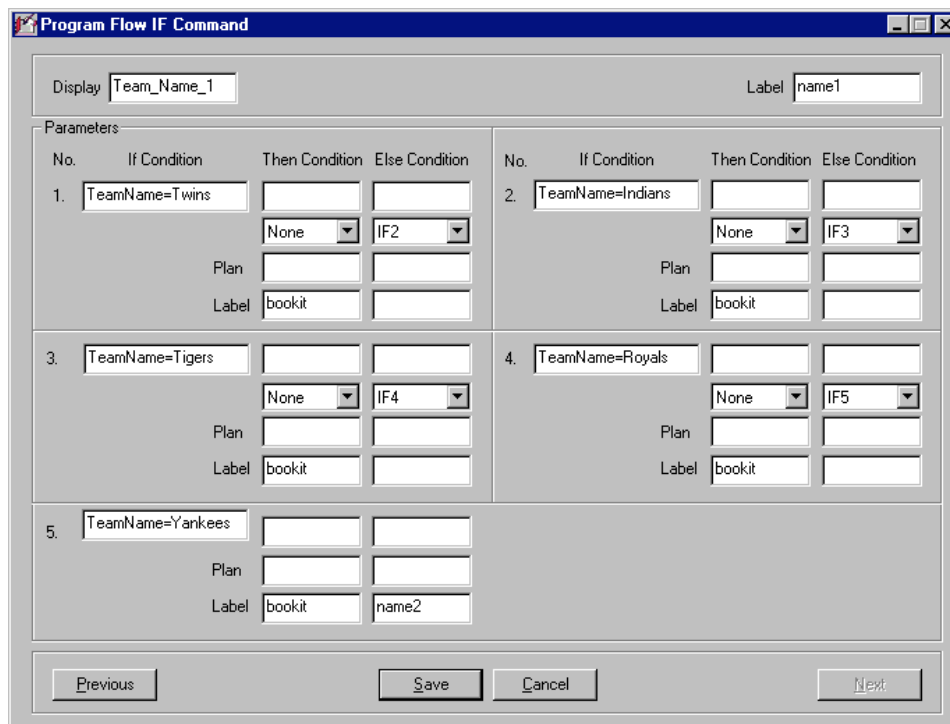


Figure A-14. First in a Series of *IF* Commands.

In the *Select* command shown in Figure A-12, the name spoken for the team name is put in the variable *TeamName* (the values for 0 – 9, \* and # are replaced using another command as will be explained later). In the series of *IF* commands, the value for this variable is tested and when the actual name is determined, it is carried to a database command where the variable value is added to the callers account. (Since the purpose of this paper is to discuss Matrix<sup>2</sup> Speech Recognition, the entire flow of this Plan will not be discussed as that is a Plan Editor topic).

To put the proper values in the *TeamName* variable for the 0 – 9, \* and # choices, each of these is sent to a Label with the two word team name and assigned the correct value. This is done using a *SetVar* command as shown in Figure A-15.

Parameters			
Variable Names	Variables Values	Variable Names	Variables Values
TeamName	Red Sox		

Figure A-15. Setting Correct TeamName with SetVar

Each *SetVar* command is followed by a *GoTo* command to take the flow to the first *IF* command – *name1*.

### GetDTMF

The *GetDTMF* command behaves in a similar manner to the 'Voice Recognition' portion of the *Select* command. As in the *Select* command, any retrieved word, number or series of numbers is stored in the 'Store DTMF In' variable. The value that is stored in this variable can then be retrieved and acted upon by another command.

In the *Select* command you can store valid words in the 'Voice Recognition' field. At the time of this writing, the only anticipated values are numbers – either single digit or number strings such as PIN's or credit card numbers. It is possible that future development efforts will add this 'Voice Recognition' field to the *GetDTMF* command if it is deemed necessary. For the time being, however, any word that is contained in the Grammar Lexicon (refer to the *ASRLexMan* section of this paper) is valid for the *GetDTMF* command. Foreseeable uses are the spelling out of an e-mail address or gathering street address, city, state and country information. Using the *ASRLexMan* utility will allow these options to be achieved.

#### **IV. Summary**

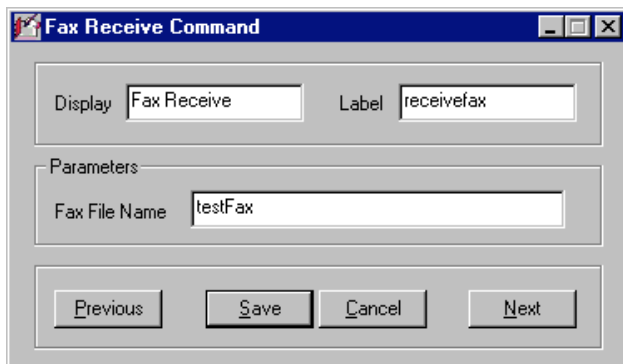
The new Matrix<sup>2</sup> Speech Recognition feature adds a tremendous advantage to the already powerful Matrix Plan Editor. Some examples have been shown here, but the only limits are one's imagination. It has been shown that ASR adds much more power and flexibility to the *Select* command providing more options to the caller in a less cumbersome way. More information can now be collected in a single call that goes way beyond the limitations of a touch-tone keypad.

## B. MCCT's Matrix<sup>2</sup> Fax Application Using Aculab Resources

### I. Introduction

MCCT has chosen the Aculab Prosody platform to be the foundation for its Matrix<sup>2</sup> Fax application. Features of the Matrix<sup>2</sup> Fax application include the ability to send and receive faxes, broadcast fax and fax-on-demand. The fax framework uses a .tiff image (Group 3 TIFF) for both transmit and receive; however, the fax source may be .txt, .doc, .rtf (and many more) that may be converted to a .tiff image.

In the Matrix<sup>2</sup> Plan Editor, all that is required for Fax commands is a simple set of the basic receive and send commands to invoke the Aculab Fax resources. The Plan Editor already contains a robust set of commands that can handle the remaining aspects of any fax application such as setting up a fax broadcast and selecting fax titles to be sent in a fax-on-demand scenario. The FAXRECEIVE and FAXSEND commands are shown in Figures B-1 and B-2, respectively.



The screenshot shows a dialog box titled "Fax Receive Command". It has a blue title bar with standard window controls. The dialog is divided into sections. The top section has two text boxes: "Display" containing "Fax Receive" and "Label" containing "receivfax". Below this is a "Parameters" section with a "Fax File Name" text box containing "testFax". At the bottom, there are four buttons: "Previous", "Save", "Cancel", and "Next".

Figure B-1. FAXRECEIVE Command Screen.



The screenshot shows a dialog box titled "Fax SEND Command". It has a blue title bar with standard window controls. The dialog is divided into sections. The top section has two text boxes: "Display" containing "Fax Send" and "Label" containing "sendfax". Below this is a "Parameters" section with five "Fax File Name" text boxes. The first box contains "TestFax.tif", and the others are empty. At the bottom, there are four buttons: "Previous", "Save", "Cancel", and "Next".

Figure B-2. FAXSEND Command Screen.

In this paper, there are discussions on Aculab Prosody DSP resources required, fax applications using the Matrix<sup>2</sup> Plan Editor and .tiff format usage. The thrust is the integration of the fax resources with the capabilities of recording and sending voice messages and sending e-mail; in essence, these are the components of Unified Messaging. (The next White Paper will be on Unified Messaging once it is formally packaged in the Plan Editor with special commands and supporting documentation). All the pieces are in place; it is just a matter of formal presentation and appearance.

## II. Matrix<sup>2</sup> Aculab Fax Overview

The Matrix<sup>2</sup> system has historically run on RedHat Linux 7.2, kernel 2.4.20-28.7. The Aculab software ran on dtk132, which did not support Aculab Fax. Aculab was upgraded to dtk141 (now dtk142b4), as this version was developed to support both Fax and VoIP. After this upgrade, there were problem in compiling using the gcc2.96-98 compiler – in fact, there were problems compiling with other compilers such as gcc2.96-112. It was decided to upgrade RedHat to version 7.3 since a later version would not have some of the bugs in the compilers. We decided to install Aculab dtk141 on RedHat Linux 7.3 with the gcc2.96-110 compiler and Fax compiled as expected.

### Prosody Resources – General

Aculab's voice resource boards are made up single, dual or quad T-1/E-1 modules with additional Sharc modules available: there can be 1, 2, 3 or 4 Sharc modules on a board – each providing a maximum of 64 voice resources, depending on the features used. For a quad board with 96 channels for T-1 and 120 channels for E-1 there are criteria for determining how many voice resources are required. These are (assume quad boards):

<u>Feature</u>	<u>Resources per Sharc</u>
Play/Record w/DTMF	64
Conferencing	64
Speech Recognition and DTMF	64
Speech Recognition w/Barge-in & DTMF	25
<b>Fax Send</b>	<b>46</b>
<b>Fax Receive</b>	<b>12</b>

For a system with 96 ports (92 assuming ISDN where the D-channel is not counted) being used for fax and standard Play & Record, 276 resources would be required if the system were full and every feature was being used simultaneously (This is calculated based on three features per channel, i.e., 1. Play OR Record, 2. fax send and 3. fax receive → 3 x 92 = 276).

Since not all features will ever be used simultaneously, fewer resources are required. As an example, when a caller is sending a fax he is not receiving one, and vice versa. The more accurate estimate of maximum resources used at a given time is 184. In this situation, due to the limited resources provided from one Sharc for fax send & receive, more sharcs will be required than in a simple voice resource environment. Based on the "Resources per Sharc" numbers shown on the previous page and assuming that any given time there could be a maximum of 92 voice resources (2 Sharcs = 128), 12 fax receive (1 Sharc) and 46 fax send (1 Sharc) you would need 4 Sharcs to provide the required resources. For voice resources, 2 Sharcs provide 36 more resources ( $128 - 92 = 36$ ) than required for a full system so these could provide a buffer for the 12 fax receive and 46 fax send estimates. If more fax activity than the estimates was encountered, more Sharcs would need to be added and since there is a maximum of four Sharcs per quad board and additional board would be required which would not be cost-effective unless more ports were added. The management of resources will require good planning for a cost-efficient system. **Note:** Typically, for 92 ports, 12 fax receive and 46 fax send are more than adequate for any given peak time.

### Fax Firmware Requirements

For each Aculab feature being used, there are firmware modules that must be loaded when the system is booted. The specific modules are edited into the "LoadSharc.sh" shell script as will be shown. The modules are in the form of .elf files as shown below:

```
[root@linux9 /]# cd /usr/aculab/dtk141/ting/sharc/gen/
[root@linux9 gen]# ls
```

```
11_to_8.elf fast.elf passthru.elf recA.elf timerx.elf
8_to_11.elf fskasyrx.elf play16b.elf recima.elf tonegen.elf
ansdet.elf fskpll.elf play8b.elf recms8b.elf v110.elf
asyrx.elf fskrx.elf playabl.elf recmu.elf v110rlpr.elf
asytx.elf fsktx.elf playA.elf recoki.elf v110rlpt.elf
civ.elf gainbg.elf playima.elf six2five.elf v17tx.elf
conf.elf grunt.elf playms8b.elf sixkin.elf v27rx.elf
cpumon.elf hdlcrx.elf playmu.elf sixkout.elf v27tx.elf
cwrx.elf hdlctx.elf playoki.elf slow.elf v29rx.elf
cwtx.elf inchan.elf prefsuf.elf sync.elf v29tx.elf
datarx.elf kernel.elf rec16b.elf syncrx.elf
datatx.elf kernel.smf rec8b.elf syncctx.elf
echocan.elf outchan.elf recablk.elf td.elf
```

The required modules for any Aculab feature can be determined by using the Aculab TiNG Channel Capacity Calculator; a list of software modules is given once all of the feature requirements are filled in. The required modules can also be determined manually using the Prosody Software Modules tables, shown on the next few pages.

## Prosody software modules

Prosody uses modular firmware, with each module performing a function. This allows the exact mix of firmware to be downloaded.

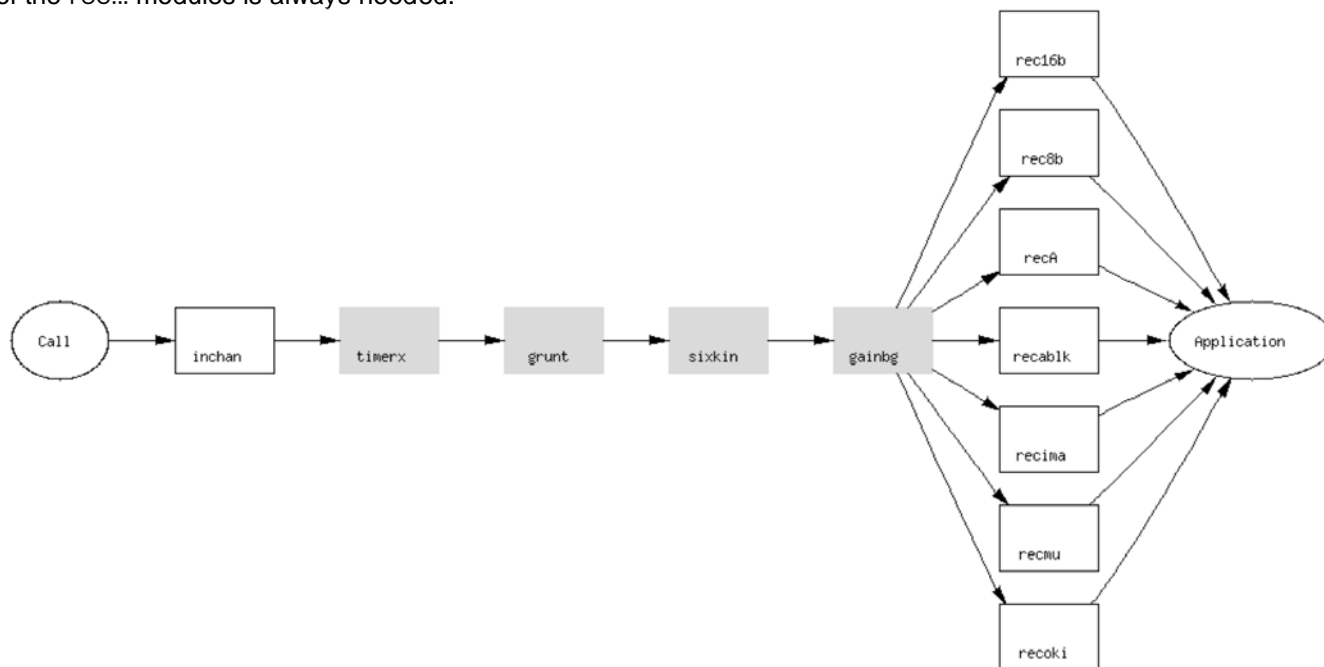
## Data communications

Data Communications modules	
Module	Purpose
asyrx	Async receive. This is used to implement the async encoding in a receive protocol. It decodes a stream of digital data into a sequence of characters.
asytx	Async transmit. This is used to implement the async encoding in a transmit protocol. It encodes a sequence of characters as a stream of digital data.
cwrx	CW modem receiver. This decodes an incoming signal from a CW modem. However it does not convert it to digital data, so either <a href="#">fskpll</a> or <a href="#">fskasyrx</a> must be used, depending on how the data is encoded, to complete the decoding.
cwtx	CW modem transmitter.
datarx	Raw data receiver. This is used to run a protocols directly on a digital bearer channel (in contrast to speech and modems which encode the data as sounds).
datatx	Raw data transmitter. This is used to run a protocols directly on a digital bearer channel (in contrast to speech and modems which encode the data as sounds).
fskasyrx	Async receiver for FSK modems. This decodes a sequence of characters from the data received by an FSK modem receiver. This is different from running async on other received data streams because there is no clock.
fskpll	Reconstruct a clock from a received FSK data stream. If the output of an FSK modem receiver is to be interpreted as a synchronous encoding, then the data must be divided into bits using a clock recovered from the signal. This module does this, converting the received signal into a digital data signal suitable for decoding by HDLC or sync.
fskrx	FSK modem receiver. This decodes an incoming signal from an FSK modem. However it does not convert it to digital data, so either <a href="#">fskpll</a> or <a href="#">fskasyrx</a> must be used, depending on how the data is encoded, to complete the decoding.
fsktx	FSK modem transmitter.
hdlcrx	HDLC receiver.
hdlctx	HDLC transmitter.
six2five	A helper module used by the modem transmitters V.27ter, V.29, and V.17.
syncrx	Synchronous data receiver.
synctx	Synchronous data transmitter.
v110	The V.110 protocol.
v110rlpr	The V.110 RLP receiver.
v110rlpt	The V.110 RLP transmitter.
v27rx	V.27ter modem receiver.
v27tx	V.27ter modem transmitter. Also requires <a href="#">six2five</a>
v29rx	V.29 modem receiver.
v29tx	V.29 modem transmitter. Also requires <a href="#">six2five</a>
v17tx	V.17 modem transmitter. Also requires <a href="#">six2five</a>
prefsurf	Prefix/suffix provider. Adds a prefix and a suffix to each transmission. The prefix and suffix can be any number of bits in length.

Data communications modules must be used in the combinations described in [Prosody data communications Protocols and Encodings](#).

## Speech recording

This diagram shows the relationship between the modules needed for recording. Those shaded are optional. One of the rec... modules is always needed.

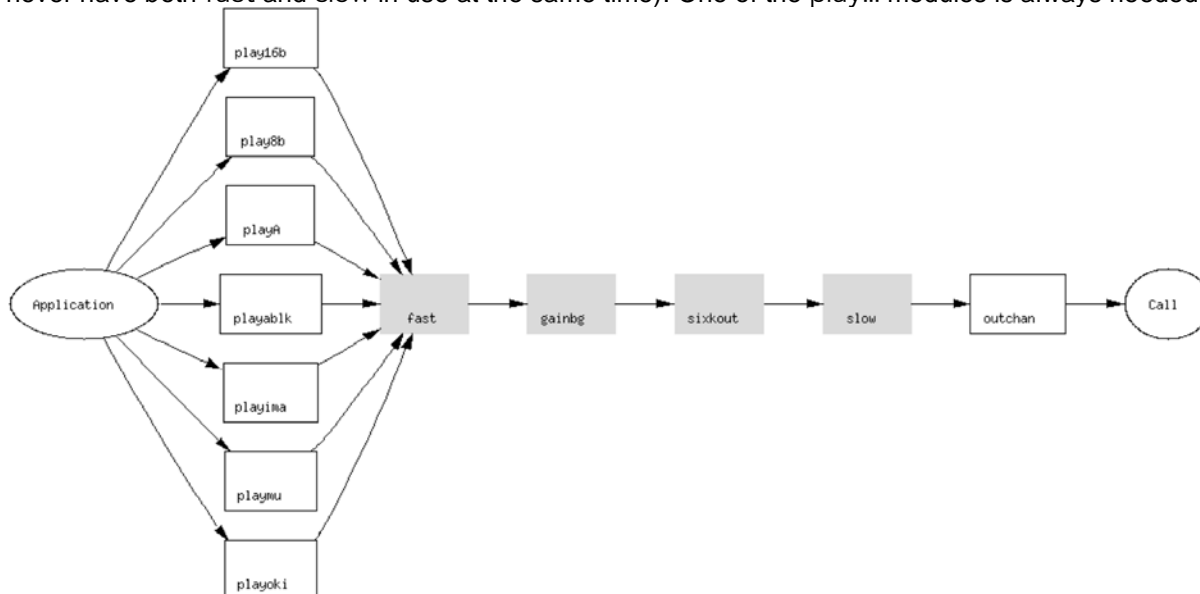


Speech input modules	
Module	Purpose
inchan	Generic companded input. This is required for any form of processing on incoming data which is a speech or analogue signal. The only input which does not need it is digital data communications. If this module has not been downloaded, no recording, conferencing or detection will work.
Speech recording modules	
Module	Purpose
gainbg	Volume control, automatic gain control, and addition of a background signal. These can all be used with <a href="#">sm_replay_start()</a> or <a href="#">sm_replay_adjust()</a> .  This module is also used to implement the volume control and automatic gain control facilities used by <a href="#">sm_record_start()</a> .
rec16b	Record 16-bit linear data.
rec8b	Record 8-bit linear data.
recms8b	Record 8-bit linear data with offset of 128 (Microsoft .WAV 8-bit).
recA	Record A-law companded data.
recablk	Record data encoded in ACUBLK format.
recima	Record data encoded in IMA format.
recmu	Record mu-law companded data.
recoki	Record data encoded in OKI format.
sixkin	Converter to allow recording at 6kHz instead of 8kHz. This is used in conjunction with a recording format to reduce the size of the recorded data by 25%.
timerx	Timer to provide a limit on recording duration.

Note: all speech recording modules require [inchan](#) as well.

## Speech playing

This diagram shows the relationship between the modules needed for playing. Those shaded are optional (and you would never have both fast and slow in use at the same time). One of the play... modules is always needed.



Speech output modules	
Module	Purpose
outchan	Generic companded output. This is required to produce nearly any form of speech or analogue output signal. The only outputs which do not need it are tone generation and digital data communications. If this module has not been downloaded, no replay or conferencing will work.
tonegen	Tone generation. This is used by <a href="#">sm_play_tone()</a> , <a href="#">sm_play_cptone()</a> , and <a href="#">sm_play_digits()</a> .

Speech playing modules	
Module	Purpose
fast	Replay at speeds over 100%. The speed is selected by <a href="#">sm_replay_start()</a> or <a href="#">sm_replay_adjust()</a> . If this module has not been downloaded, the replay will play at normal speed.
gainbg	Volume control, automatic gain control, and addition of a background signal. See <a href="#">gainbg</a> listed under speech recording.
play16b	Play 16-bit linear data.
play8b	Play 8-bit linear data.
plays8b	Play 8-bit linear data with offset of 128 (Microsoft .WAV 8-bit).
playA	Play A-law companded data.
playablk	Play data encoded in ACUBLK format.
playima	Play data encoded in IMA format.
playmu	Play mu-law companded data.
playoki	Play data encoded in OKI format.
sixkout	Converter to allow playing of 6kHz signals at 8kHz. This is used in conjunction with a replay to play data which was recorded as a 6kHz variant of a format normally used at 8kHz. Such a variant is used to reduce the size of the recorded data by 25%.
slow	Replay at speeds below 100%. The speed is selected by <a href="#">sm_replay_start()</a> or <a href="#">sm_replay_adjust()</a> . If this module has not been downloaded, the replay will play at normal speed.

Note: all speech playing modules require [outchan](#) as well.

## Detection

Detection modules	
Module	Purpose
grunt	Grunt detection and silence elimination. Used by <a href="#">sm_record_start()</a> for silence elimination in recordings and by <a href="#">sm_listen_for()</a> . If this module has not been downloaded, no grunt will be detected and recordings will not have silences removed.
td	Tone detection. Used by <a href="#">sm_listen_for()</a> for tone detection and by <a href="#">sm_record_start()</a> for tone suppression in recordings. If this module has not been downloaded, no tones will be detected and tones will remain in recordings.

## Conferencing

Conferencing modules	
Module	Purpose
conf	Conferencing. This is required if you want to start a conference (which is done using <a href="#">sm_conf_prim_start()</a> ). If this module has not been downloaded, any attempt to start a conference will produce an error.

## Echo cancellation

Echo cancellation modules	
Module	Purpose
echocan	Echo cancellation. If this module has not been downloaded, an attempt to use echo cancellation will use an unmodified signal.
passthru	A module which permits an incoming signal to be directed to an output. This is used by echo cancellation to output the processed signal. It is not needed if the processed signal is only being recorded.

## Signal categorization

Signal categorization modules	
Module	Purpose
ansdet	Answering machine detection. Used by <a href="#">sm_catsig_listen_for()</a> to distinguish between a live speaker and an answering machine.

## Test and debug

Test and debug modules	
Module	Purpose
sync	Synchronizer. Allows the start of certain operations to be synchronized. This is used for testing where it is useful to record several signals with a known relationship. The implementation of this facility relies on some features which are liable to change between versions.
cpumon	Cpu usage monitor.

Using the above information to group modules into categories plus knowing what modules are needed from the calculator or the tables above, the script structure for the "LoadSharc.sh" shell script can be formulated as shown on the next pages.

LoadModule (./lm) portion of the LoadSharx.sh Shell Script for the first Sharx:

```
##### Board 1 #####
T1board="-c Prosody_156799 -m 0"
echo "Load Modules T1 Board Sharx 0 $T1board"
```

**(This column not in Shell Script – Info Only)**

	<u>Category from Prosody Software Module Sheet</u>	
<b># play</b>		
./lm outchan \$T1board	<b>Speech Output Module</b>	
./lm gainbg \$T1board	<b>Speech Playing Module</b>	
./lm playA \$T1board	"	"
./lm playmu \$T1board	"	"
./lm playoki \$T1board	"	"
./lm playablk \$T1board	"	"
./lm fast \$T1board	"	"
./lm slow \$T1board	"	"
./lm sixkout \$T1board	"	"
<b># record</b>		
./lm inchan \$T1board	<b>Speech Input Module</b>	
./lm recA \$T1board	<b>Speech Recording Module</b>	
./lm recmu \$T1board	"	"
./lm recoki \$T1board	"	"
./lm recablk \$T1board	"	"
./lm sixkin \$T1board	"	"
./lm timerx \$T1board	"	"
<b># detections</b>		
./lm td \$T1board	<b>Detection Module</b>	
./lm grunt \$T1board	"	"
<b># generation</b>		
./lm tonegen \$T1board	<b>Speech Output Module</b>	
./lm ansdet \$T1board	<b>Signal Categorization Module</b>	
<b># fax services</b>		
./lm fskrx \$T1board	<b>Data Communications Module</b>	
./lm fsctx \$T1board	"	"
./lm v27tx \$T1board	"	"
./lm v27rx \$T1board	"	"
./lm v29tx \$T1board	"	"
./lm v29rx \$T1board	"	"
./lm v17tx \$T1board	"	"
./lm fskpII \$T1board	"	"
./lm hdlctx \$T1board	"	"
./lm hdlcrx \$T1board	"	"
./lm synctx \$T1board	"	"
./lm syncrx \$T1board	"	"
./lm six2five \$T1board	"	"
./lm datatx \$T1board	"	"
./lm datarx \$T1board	"	"

**# continuous word speech recognition**

#./lm echocan \$T1board

**Echo Cancellation Module**

#./lm passthru \$T1board

" "

**# conferencing**

#./lm conf \$T1board

**Conferencing Module**

Once all of the correct modules are loaded for the desired features, the application can perform as desired. The modules are used in conjunction with the function calls and event handlers that the C/C++ programmer sets up in the code. This aspect is beyond the scope of this paper.

**III. Integration of Fax in the Matrix<sup>2</sup> Platform**

There are many Matrix<sup>2</sup> commands that perform a variety of functions in a Plan (or application). Looking at some of the Command Groups, we can see that there are pre-existing commands that perform the functions required for Unified Messaging but are not labeled as such (yet). A sampling of these includes:

<b><u>Command Group</u></b>	<b><u>Command</u></b>	<b><u>Function</u></b>
<u>General</u>	PLAYMSG	Plays a voice file to caller in an appropriate format
	RECORDMSG	Records caller's message in .wav file format
	LINUXCMD	Can be used for e-mail and other Shell Scripts as needed
<u>Program Flow</u>	IF	Tests for conditions such as Fax or Voice
	SELECT	Gives options for callers to choose from; this can be via DTMF or Speech Recognition
<u>Telecom</u>	DIAL	Allows the caller to dial out to another telephone number from the inbound call
	GETDTMF	Allows caller to enter digits via the touchtone keypad or to enter numbers or word strings using Speech Recognition

<u>Command Group</u>	<u>Command</u>	<u>Function</u>
<u>Text-to-Speech</u>	FILETOFILECONVERT	Converts text such as e-mail to an appropriate voice file (.wav or other)
	VARTOFILECONVERT	Converts text in a User Variable (e-mail, web page text, etc.) to an appropriate voice file
<u>Fax</u>	FAXRECEIVE	Receive fax files in tiff format and assign a filename
	FAXSEND	Send one or more fax files in broadcast or on-demand mode

There are numerous Matrix<sup>2</sup> commands that make up a Plan (application), however the ones in the previous table are those that will be used in a Unified Messaging (UM) environment until formal UM commands are created. The LINUXCMD is versatile in that any shell script can be written in Linux command line format to accomplish a wide variety of functions such as sending e-mail and connecting to a web-site.

The next White Paper will be on the subject of Unified Messaging. In this paper, it has been shown that the pieces are in place. MCCT's next step is to develop the formal UM commands with supporting documentation so that there is a more streamlined appearance for the person writing the Plan or Application. There will be a new Command Group with potential commands such as:

CHECKEMAIL – specifically search for unopened e-mails in an account;

CHECKFAX – check for new faxes;

CHECKVOICE – look for new voice mail messages;

CONVERT – Text-to-Speech commands with added features;

SENDEMAIL – this will include functions such as CC, Attach, etc.;

SENDFAX – reuse the existing FAXSEND command;

SENDVOICE – Initiate, forward or reply to a voice message.

Of course, these will be discussed and determined by the design team and the commands may appear differently than shown here; the purpose is to satisfy the requirements of Unified Messaging as used by the community - this includes web-interface.

## V. Summary

The addition of Aculab Fax to the MCCT Matrix<sup>2</sup> platform provides the missing piece for Unified Messaging on an IVR. All of the functionality exists and Plans are currently written that use all of the UM functions. MCCT can write custom Plans or Applications that will meet your detailed requirements.

## C. MCCT's Matrix<sup>2</sup> Conferencing Using Aculab Resources

### I. Introduction

Historically, MCCT has provided Conferencing Applications to its customers for the past ten or more years. Initially, MCCT conferencing relied on Dialogic MSI boards to provide the conferencing resources and as technology advanced, changed to the Dialogic DCB series of boards. While the conferencing functionality and performance was good, there was room for improvement. Conference sizes were limited to a maximum of 32 conferees and with more people in the conference, the voice quality was not as clear as with smaller numbers in the conference.

Through extensive research, MCCT discovered Real Time Systems, Inc. (RTSI) who had an ISA conferencing board series – The Vendetta boards. These conferencing boards allowed for larger conference sizes and no degradation of voice quality and were soon replacing the Dialogic boards. MCCT conferencing applications, as a result, had significant improvement in the voice quality (echo cancellation) and in the flexibility of conference sizes with no degradation in larger conferences.

The Dialogic and Vendetta boards were ISA boards using the Signal Computing System Architecture (SCSA) bus. The SCSA bus encompasses:

- ISA slot plug-in;
- 1024 time slots;
- 2 MB throughput;
- Clocks from 1<sup>st</sup> span and
- 26-pin bus.

The Aculab boards are PCI High-density and use the H.100 architecture that supports:

- PCI slot plug-in;
- 4096 time slots;
- 8 MB throughput;
- Clock redundancy and
- 68-pin bus.

It can be easily seen from the above comparisons that the H.100 bus architecture is four times faster and has quadruple the capacity of the SCSA bus. This results in much clearer and crisper voice quality for MCCT's Matrix<sup>2</sup> Conferencing. While the older versions of conferencing are still supported, the new Aculab platform is MCCT's plan for new systems.

## II. Hardware Overview

Aculab's voice resource boards are made up single, dual or quad T-1/E-1 modules with additional Sharc modules available: there can be 1, 2, 3 or 4 Sharc modules on a board – each providing a maximum of 64 voice resources, depending on the features used. For a quad board with 96 channels for T-1 and 120 channels for E-1 there are criteria for determining how many voice resources are required. These are (assume quad boards):

<u>Feature</u>	<u>Resources per Sharc</u>
<b>Play/Record w/DTMF</b>	<b>64</b>
<b>Conferencing</b>	<b>64</b>
<b>Speech Recognition and DTMF</b>	<b>64</b>
Speech Recognition w/Barge-in & DTMF	25
Fax Send	46
Fax Receive	12

For a system with 96 ports (92 assuming ISDN where the D-channel is not counted) being used for conferencing, standard Play & Record with DTMF and Speech Recognition and DTMF, 276 voice resources would be required if the system were full and every feature was being used simultaneously (This is calculated based on three features per channel, i.e., 1. Play OR Record, 2. Conferencing, 3. Speech Recognition & DTMF →  $3 \times 92 = 276$ ). Since not all features will ever be used simultaneously, fewer resources are required. As an example, when a caller is in a conference, Speech Recognition is not be used and vice versa. The more accurate estimate of maximum resources used at a given time is 184. In this situation, three Sharcs equaling 192 voice resources is a better estimate, but four Sharcs would provide a comfort zone where one Sharc could be dedicated to conferencing only and conferencing could be shared with other features on another Sharc. This would effectively provide a dedicated 60 to 64 resources on one Sharc and 30 to 32 on a shared Sharc. The Matrix<sup>2</sup> daemon looks at the Sharc "load module" log for each Sharc so that these modules can be used in the Conferencing configuration table(s). All resources are shared on the H.100 bus.

Now that the resources required for a specific system and its applications have been described, the Matrix<sup>2</sup> Aculab conferencing capabilities can be addressed. The voice resource discussion above is included to demonstrate the architecture of the Matrix<sup>2</sup> conferencing model.

The Matrix<sup>2</sup> conferencing commands are included in the Matrix<sup>2</sup> User Manual in Chapter 4, which is appended to this document. The remainder of this white paper describes the Matrix<sup>2</sup> Aculab Conferencing features, commands and functionality. You may reference Matrix<sup>2</sup> User Manual, Chapter 4 for more detailed descriptions of each command.

### **III. Conferencing Commands and Features**

#### **A. Command Overview**

There are eight commands in the Conferencing Command Group. These are: *EAVESDROPCONF*, *GETFREECONF*, *NAMECONF*, *NUMOFCONFEREES*, *PLAYINCONF*, *PLAYTONECONF*, *PUTINCONF* and *SC10N1*.

EAVESDROPCONF. This command, given its name, is an administrative command for supervisors and monitors only. Its purpose is to allow for training of new operators, ensure that customers are treated properly or some other function of the host company. This is a "listen only" command and the monitor may not be heard.

GETFREECONF. When callers first dial in, they can be placed in the first available conference as determined by this command. It defines what an empty conference is (with or without an Operator/Agent), whether an Operator/Agent is required and if the caller must fall into a specific group or is eligible for all conferences.

NAMECONF. The only purpose for this command is to provide a unique name for each conference; this is useful in reviewing reports on conference activity so that statistics relate to individual conferences rather than the total of all conference proceedings.

NUMOFCONFEREES. As a complement to the NAMECONF command, this command allows for the recording of several statistics, including the number of conferees in each conference, for creating reports. Information that can be set up with this command includes: Total Operators, Total Agents, Total Seats in Use, Caller Category, Supervisors, Number of Eavesdropping Callers and Normal Conferees. Please refer to Appendix A for more a detailed description of these titles.

PLAYINCONF. This is a very straightforward command that allows for playing messages to the caller before entering a conference and/or playing messages to the callers in the conference. Messages to the caller outside the conference can be in the form of announcements ("You are about to enter conference xyz") and messages to the callers in the conference can be announcing a new caller, ads or whatever is appropriate.

PLAYTONECONF. Instead of playing a message to announce a caller as discussed in the PLAYINCONF command, this command allows for a tone to be played on entry to any conference. Tones for entrance and exit may also be called on with the PUTINCONF command, but this command is universal for all conferences.

PUTINCONF. This is the workhorse command of Matrix<sup>2</sup> Conferencing and is the command that sends the caller to the destination conference. With this command, callers can be categorized with attributes such as coach, pupil, muted or in a broadcast setting. A caller can also have special flags including normal, agent, operator or supervisor. When a caller is alone in a conference, the determination of music being played or not is made here. Entrance and exit tones (to the conferees) are also decided here. Detailed descriptions of each function in this command can be found in Appendix A.

SC1ON1. This is another versatile command that allows two conferees to speak privately in a two-party conference. Some of the features include storing the port number of the other caller, setting up messages, determination of music messages, wait times and exit keys.

Figure C-1 is a flow chart showing a sample of how these commands work together in a conferencing Plan.

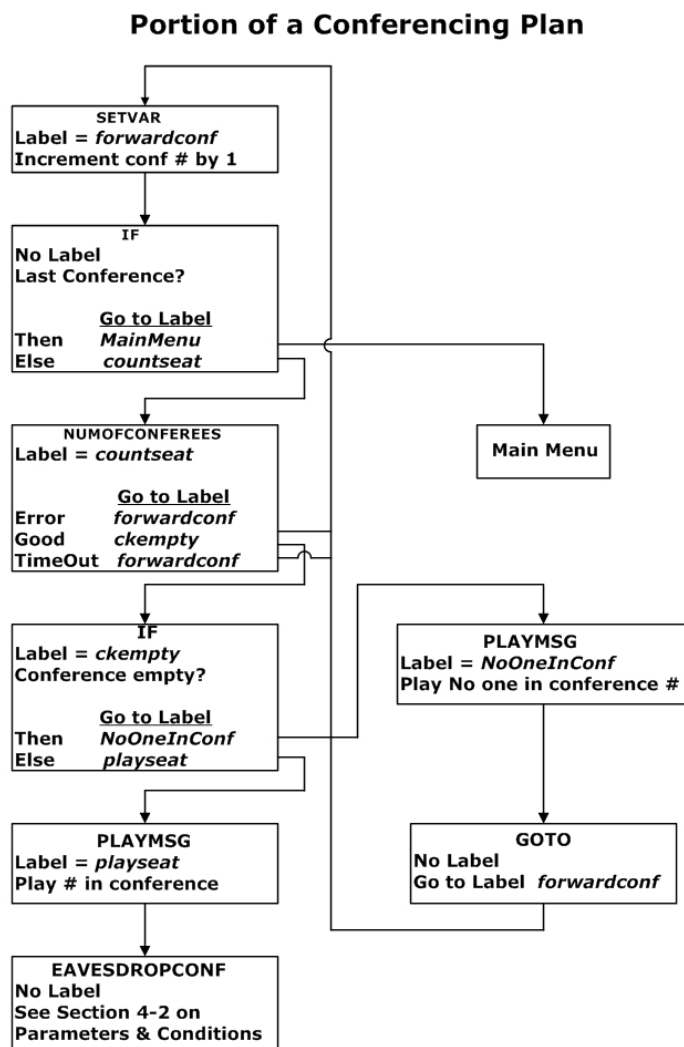
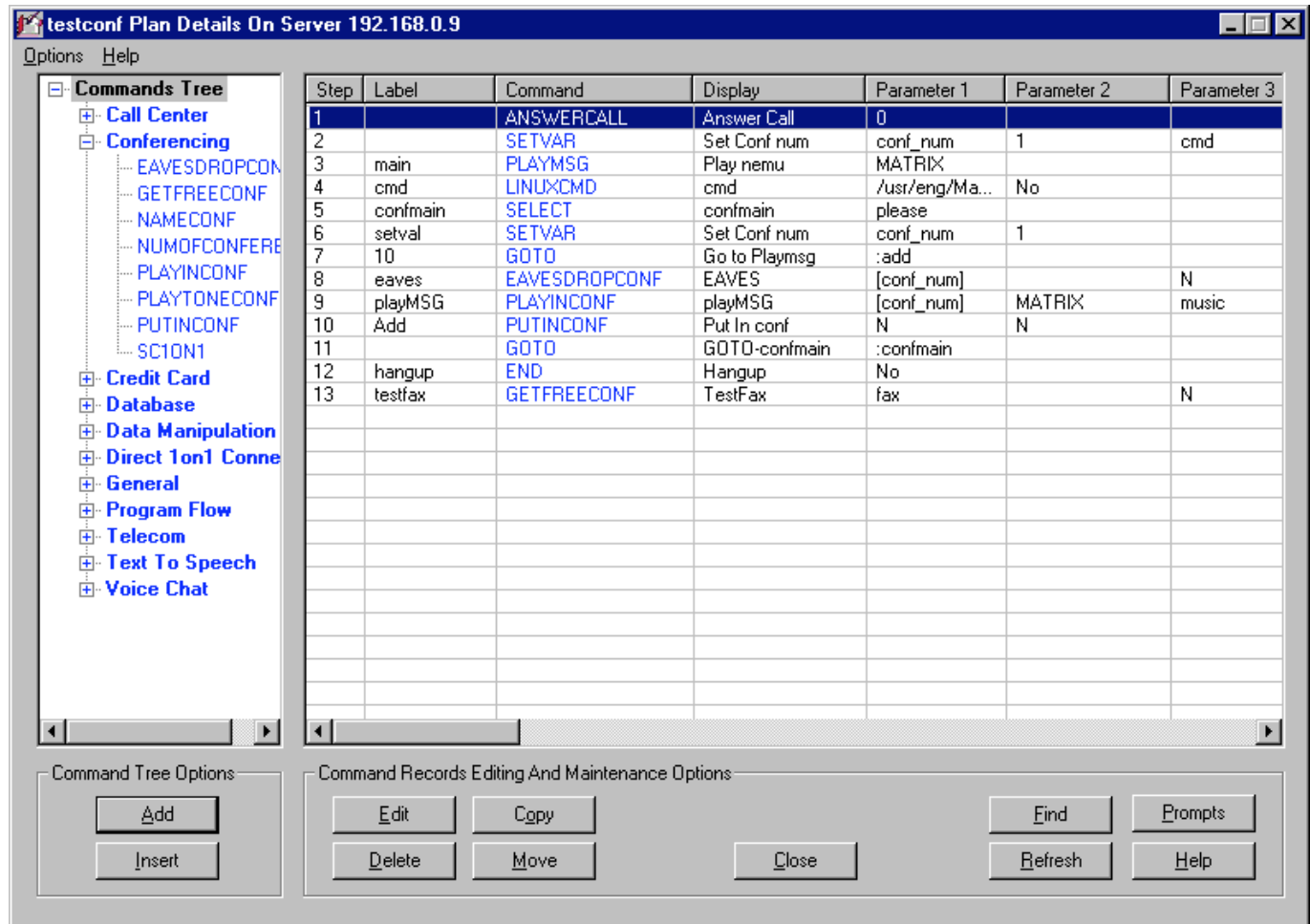


Figure C-1. Sample Flow Chart of a Conferencing Plan.

B. Command Structure

All Matrix<sup>2</sup> commands are based on MySQL Tables. In the Plan Editor, There are 22 Parameter columns where some contain values, expressions, Linux commands or variables. These parameters map directly to command tables in the “matrixplandb” database. Figure C-2 shows the Plan Editor screen where the first three parameter columns may be seen.



**Figure C-2. Plan Editor Screen with First 3 Parameter Columns.**

These parameter columns correlate to columns in the MySQL database tables and are considered the “back-end” data structure. The table structures are set independently of any command screen. Since the database source must work for all commands and their corresponding screens, there is no one-on-one correlation in the mapping. Each command will have a different mapping scheme.

Figure C-3 shows the PLAYINCONF command screen where Parameters 1, 2 and 3 from the Plan Editor screen in Figure C-2 can be mapped to the fields in the command screen.



Figure C-3. PLAYINCONF Command Screen.

**IV. Summary**

The MCCT Matrix<sup>2</sup> Conferencing application is a unique combination of resources for fluid conferencing channels on a Linux/Aculab platform and easy-to-use commands based on a MySQL foundation. Both Linux and MySQL are open source, non-proprietary software environments that adds reliability and high performance since neither is competitive. Linux has been proven to have high security standards and MySQL is limitless in its ability to handle huge volumes of data in a streamlined data flow.

MCCT has used these powerful tools in combination with Aculab’s feature-rich and highly reliable voice boards to result in a Conferencing application that compares to no other. The uses for MCCT’s Matrix<sup>2</sup> Conferencing are only limited by one’s imagination. For more information, contact MCCT at (800) GO 4 MCCT in the domestic US, (603) 524-2214 internationally or at [info@mcct.com](mailto:info@mcct.com).

## D. Major Feature Comparisons of Typical Competition vs. MCCT IVR Platform

Competition	MCCT
1. Non-Redundant 250 Watt Power Supplies	1. Quad Redundant Hot-Swappable Power Supplies (4x300 = 1200 Watt)
2. Single Drives using IDE	2. Redundant Hot-Swappable SCSI Hardware mirrored RAID I 36 - 73 GB Drives expandable to RAID V (Up to 2 Terabytes of storage)
3. Desktop PC Design with little or no expansion slots standard PC Backplane design - non-commercial.	3. 20 slot passive Backplane system - commercial with CPU board for Matrix <sup>2</sup> Platform. Up to five Quad T1/E1 in one IVR.
4. No network interface	4. 10/100/1000Base-T auto-switching with TCP/IP & IPX compatible network interface
5. Unproven operating system	5. SCO UNIX & RedHat Linux used by Telecomm industry and Internet Service Providers for over a decade (Internet backbone)
6. No or minimal Tape Backup	6. 10 GB uncompressed Tape Backup with Disaster Recovery software - recover within a few hours versus rebuild & lose applications & reports.
7. No Reboot device	7. External Remote web-enabled Reboot device independent of operating system.
8. Tech Support 9AM to 5PM Weekdays and a Return & Repair policy resulting in lengthy downtime.	8. With 1-year full warranty, 24/7 Tech Support, 10 minute response times plus our advance replacement policy with minimal downtime; yearly extensions are available.
9. Unknown number of countries where systems installed & unknown number of ports.	9. Installations in 68 countries and over 500,000 ports installed and running. Demonstrates our vast experience in handling a multitude of international protocols.
10. Typically IDE bus for drives and peripherals which does not support RAID	10. Use Ultra-320 SCSI bus; all drives are hardware mirrored with RAID I and for large voice storage, RAID V arrays are available with up to 2 Terabytes.
11. Limited expansion capability due to PC environment and operating system limitations.	11. In one chassis, up to 20 E-1's or 20 T-1's inbound. Expansion is further enhanced by using SCSI bus architecture and the RAID V option so that up to 480 ports are attainable in one integrated system.

12. No uninterrupted power protection included and no auto-shutdown software.	12. Integrated rackmount UPS with auto-shutdown software included with each system.
13. No SNMP remote alert capability.	13. MCCT offers optional SNMP Version 1 hardware and software MIBS manager. All parameters on the system (heat, fans, voltages, processes, drives, network connections, etc.) are monitored continuously.
14. No user application generator.	14. Full robust user application generator with GUI interface and the ability to record voice prompts or convert text to speech on the fly.
15. No conferencing operator interface.	15. Full drag and drop web-enabled conference control screen with coach-pupil, muting, record/ monitor, lecturing and operator features.
16. No credit card processing.	16. Full credit card processing optional program with auto reporting and batch processing.
17. Little or no in-house test equipment.	17. Over \$ 1 million invested in test equipment and R & D per year.
18. NT based platforms susceptible to internet viruses and hackers.	18. There are no known viruses on MCCT's SCO UNIX or Linux platforms which includes automatic 'unknown user' lockout.
19. Mean-time-between-failure (MTBF) un-documented and unknown.	19. MCCT systems have MTBF of 99.9%; up time is tracked, recorded and documented.
20. No customer service interface.	20. MCCT's optional CSR package is robust, versatile and web-enabled.
21. Limited Call Progress Analysis	21. Extensive Call Progress Analysis with positive voice detection, answering machine detection, SIT tones, busy, no answer, fax and advanced ISDN messaging.
22. Synchronous process call handling. If one process dies, the entire call processing on the machine dies.	22. Asynchronous process call handling; one process for each line. If one line or process stops, the remaining lines on the IVR continue to process calls. The stopped process can be restarted locally or remotely or with user configurable parameters.

**MCCT cares about your successful and profitable business.**

**Down time is lost revenue.**

**We are there for you 24 hours a day - 7 days a week**

### **III. Sample Application(s)**

Some of MCCT's major applications will be discussed in this section. These will be described in an overview format with examples of some of the steps in the Plan(s). Flowcharts and detailed descriptions are available from MCCT by calling us at 800-GO-4-MCCT or 603-524-2214 or e-mailing us at [info@mcct.com](mailto:info@mcct.com).

Applications described herein include:

- Unified Messaging,
- Do Not Call,
- University Help Desk,
- Nationwide Mail Information,
- Nationwide Delivery Tracking Control and
- Field Personnel Protection Plan.

#### Unified Messaging

This is a customizable Plan that provides "one-stop shopping" for callers to receive and send: voice mail, e-mail and fax. The generic version uses DTMF to select options, however it can be easily modified to use Speech Recognition.

With voice mail, messages can be recorded and played back and they can be sent and received, depending on the caller's selection(s). Voice messages may also be sent as attachment to e-mails.

The e-mail input is usually via web-interface where text files may be created and sent to the IVR. E-mails can be retrieved and sent from a caller's account as well as forwarded and replied to. Any attachment can be sent or captured.

A caller may wish to retrieve faxes from a list (fax-on-demand) or broadcast faxes to one or multiple recipient(s). With the Matrix<sup>2</sup>/Aculab Fax aspect of Unified Messaging the ".tif" format is used as described in the White Paper on Fax. Most files (e.g., .jpg, .gif, .txt, etc.) can be converted into .tif format using the Matrix<sup>2</sup>/Aculab Fax engine.

There is also the option to have e-mails converted into voice files using the TTS (Text-to-Speech) feature of Matrix. It is best to first convert to a voice file rather than listen to the conversion "on the fly", i.e., real time. There are two reasons for this. The first is that the Matrix<sup>2</sup> system comes with two channels of TTS thus limiting the number of simultaneous conversions with multiple callers. TTS was never intended to be used real time – for the second reason for conversion first. In real time, the conversion occurs much slower than playing a converted voice file. Listening to the voice conversion would be distorted and delayed compared to the actual finished result.

## Do Not Call

This example has many possibilities where using a DTMF entry on completion of a call to update a global database is desired. Typically, this type of system is one where employees (e.g., telemarketers in the *Do Not Call* scenario) dial in, log in with a password and/or PIN and dial out from the IVR. The dial out numbers will be from a preset list all of the details of the call (start time, duration, DNIS, etc.) will be placed in a database in addition to a code entered by the employee. This database will be common to all IVR's in the system and will be universally available to those who have access.

In the *Do Not Call* example, the code entered will be related to the status of the called party. If there was no problem with the call, the employee may enter a "#1" which could mean "this party is interested in our product" or "#2" which could mean "some interest". On the other hand, the entry may be "#\*" which could mean "Do not call this party again" and would set a flag in the database telling all parties to refrain from calling this number. In the actual situation, if anyone were to call this party again, the company would be fined thousands of dollars according to existing laws.

While *Do Not Call* is the example given, there are many other possible situations for this type of application. Some of these possibilities are:

- Service calls;
- Appointment setting/reminders;
- Catalogue sales;
- Surveys;
- Political calls;
- Alumni calls from a University.

Whatever the choice of the dial-out application is, it can be customized to update any database desired for real time record keeping.

### University Help Desk

This is an application that is a major cost-saver to large universities and colleges. All aspects of a student's participation can be managed with this application as well as a secure means of parents to check on a student's progress. This "one-stop shopping" system can be a major time saver for the student as well as saving the university large staffing costs. For those of us who have gone through registrations, the ordeal of signing up for classes, getting the textbooks, etc. is well remembered.

The student can call into the system, login with a student ID and select from a large variety of topics. These include:

- Registration for classes based on hearing all of the choices or by correlating to a course schedule handout that has number codes for each class and time. The student can have his/her finalized registration faxed or e-mailed to receive confirmation.
- Purchasing textbooks based on the numerical code provided in the registration confirmation. The student can purchase a first book, then a second and so on until all are obtained (with the option of new or used). These can be charged against the student's established account or paid by credit card. Arrangements can then be made for pickup or delivery.
- Events and activities can include football and basketball games, special interest and social clubs, dances, concerts, sports activities and social events. The option to get sign-up information or obtain game tickets is provided and the student can make all necessary decisions here and come back later to add more.
- Housing can be researched on and off campus by choosing key words from lists of categories. Addresses and telephone numbers can be given to allow for further investigation eliminating the need to check bulletin boards, newspapers and other media. Another method that can be an option is to get into one of several conferences on the subject of housing where parties with rooms to rent and students needing rooms can discuss possible arrangements.
- Tutors may be located using lists in various subject categories where telephone numbers can be provided to make any arrangements that are necessary. The student may also enter one of a few conferences where tutors and students seeking tutors can make arrangements.
- Parents can check on student progress and financial status using a secure database path with password and/or PIN protection. This can be a very valuable feature, if provided, for the parent investing a small fortune in their child's future.

### Nationwide Mall Information

MCCT has written a Plan that makes available a database and interface to allow callers to check on any of hundreds of malls in the U.S. and any of the stores within that mall. The customer who manages all of this has been provided an easy to use interface to update the MySQL tables containing message prompts, promotions, address and telephone number information, etc.

Callers can call a main number and select one or more of hundreds of malls. Once in the mall main menu, they can search for stores by category or name and find all the information they would need such as:

- Mall and store open & closing times for each day of the week,
- Store search by name, type or category,
- Special events,
- Jobs that are available,
- Directions to malls and parking information,
- Residential information about general area,
- Call transfers to operators and stores/restaurants,
- Mall security,
- Management office,
- Maintenance,
- Public transportation,
- Sponsor information or
- Take a survey.

This is a working application and is dynamic in that the database is regularly updated. Callers planning trips can “plan ahead” if shopping is on their activity list; many people like to buy local products and can “pre-shop’ using this application. In many metropolitan areas, there are several malls that can be easily researched while sitting in the comfort of your home.

### Nationwide Delivery Tracking

Without having the extensive tracking systems used by UPS, FedEx and DHL, our customers have the ability of tracking shipments completely. This is all done with a quick telephone call from the driver. The basis for the generic application relies on a "pallet" count; however, the application may be customized to use any other measurement scheme.

At each store, e.g., Wal-Mart, Kmart, Target, etc., a unique store number is given to be entered into the system. There are pickups and/or deliveries at each store and the driver simply enters the truck number, store number, the number of pallets dropped off and the number of pallets picked up. Combined with the time the data is entered, a complete picture of each truck's route and performance is provided to the home office.

The driver may also be prompted to check in to the home office or may retrieve a voice mail message sent after his most recent stop. This adds interaction with the driver for special instructions, changes in the route or other important messages that are pertinent to the driver.

This is a simple and straightforward application that accomplishes a multitude of functionality at a very low cost.

### Field Personnel Protection

There are a variety of companies who have people in the field who go to remote sites that are unmanned. Examples include mobile phone carriers such as Cingular, Verizon Wireless, AT&T, etc.; the field personnel must go to the repeater sites and check equipment on a regular basis. There are other types of remote site checking situations, but this will serve as our example.

Historically, there have been tragedies that have occurred at some of these remote sites; these include agents being attacked and badly injured with some killed and also heart attacks incurred with no one knowing. As a result, some of these companies have required applications to be developed that would create active contact between the agent and the IVR with check in times starting a clock. If no input were received after a predetermined time then a series of events would be triggered by the IVR. These are described in this section. In addition to entering an agent ID, the agent would also enter site, switch, alarm, city and state information.

When the agent initially logs into the site with the agent ID, s/he is asked a series of questions by the IVR. The site, switch city and state information can be programmed for an automatic response in the agent's cell phone and the agent will then be asked if an alarm will be generated so that an operations center can be alerted.

Should the agent not properly log out before the pre-determined time, a series of calls will be triggered to be dialed out from the IVR that could include the regional operations center followed by legal assistance from the local authorities to conduct a search of the remote site expediently. Since some of the companies in this arena have thousands of agents, this type of application is essential based on the types of things that can occur to an individual who is in the field alone.

Additionally, there are several reports generated by the application about the daily activities of the agents. These will show trends, average times, sites in need of a visit, dial-out records and a variety of other pertinent data customizable by the system administrator.

### Special Alerts

There are many possibilities for applications that provide alerts to a variety of customers – businesses, government, individuals, medical, educational and so on. The possibilities embrace features that have already been created in different “flavors” for customers; these features include:

- Fax
- E-mail
- Voice Mail
- GIS (Geographic Information Systems)
- Conferencing

Using any or all of the features shown above, users with accounts may sign up for the many possible services that are available. The GIS feature includes zip code, telephone number, address and geographical coordinate information that is needed by law enforcement and other government customers. Conferencing can be set up by individual selection based on invitations or searches or may be automated in the event of emergencies such as amber alerts or major weather events, as examples. Some of the many available services include:

- Amber alerts
- Baseball scores
- Cancellations/Closures
- Flight information
- News
- System alarm alerts
- Traffic reports
- Weather conditions

These are just a few of the services that can be provided via e-mail, fax or voice messaging. The alerts can be triggered by events or by scheduled time slots.

## IV. Specifications

### Software Platforms

The software platforms and associated revisions are as of the writing of this manual. MCCT will ensure that compatibility will be maintained for future revisions and that revision changes will be updated in later versions of this manual.

Linux	RedHat version 7.2 & 7.3 (Kernel 2.4.20-28.7bigmem); Fedora Core 1 (Kernel 2.4) in development to support Aculab version 6.
Linux Streams Aculab	GCom LiS Version 2.13.16. Version 5, dtk142b4 (Supported); Version 6 in development.
Dialogic	Linux release 5.1, Service Pack 1 (or later).
Dialogic	Global Call.
MySQL	3.23.49.
MyODBC	Versions 2.50 and 3.51.
Fonix TTS	Version 5.1.5

### Hardware Platforms

The hardware presented in this revision of the Matrix<sup>2</sup> User Manual reflects the latest state-of-the-art equipment available. Any changes will be shown in later revisions of this manual.

#### Base Hardware Platform:

MCCT Matrix IVR  
Trenton Technologies CPU board.  
3.06 GHz Trenton XPT Dual Xeon CPU  
4 - 8 GB DRAM  
On-board:  
2 x 100/1000BaseT  
VGA  
IDE  
Serial / Parallel ports  
SCSI interface (currently un-used)  
Adaptec 2200S Ultra-320 SCSI Raid I/V Controller  
Kingston Ultra-320 Hot Swap Drive carriers  
Portwell EZ-Drive (IDE CD-ROM and Floppy combo)  
Seagate Tape Drive (IDE TR-5)  
MultiTech PCI Modem – 56K

Ports using Dialogic:

240 Ports inbound (10 T-1s/8 E-1s) or  
240 Ports outbound (10 T-1s/8 E-1s)

For the initial release, the following Dialogic and Vendetta boards are supported:

D/41ESC, D/240SC-T1, D/240SC-2T1, D/480SC-2T1D, 300SC-E1 (75 and 100 ohm versions) D/300SC-2E1 (75 and 100 ohm versions) D/600SC-2E1 (75 and 100 ohm versions), VND32, VND128, VND256.

Ports using Aculab:

480 ports inbound or 240 Ports inbound and 240 Ports outbound (or any combination where inbound > outbound)

Aculab Boards supported:

PCI Prosody Cards with up to 4 Sharc™ DSP Modules  
PCI Network Adapters  
PCI Trunk Modules  
PCI DSP/Sharc™ Modules  
Future Release: VoiP PCI Cards (Mid CY 2005)

### Programming Features and Languages

It is not required that the user be cognizant of programming languages to be able to program applications using the Matrix<sup>2</sup> Application Generator. There are, however, those who are adept at programming and would make use of the languages that are a part of the Matrix<sup>2</sup> software.

For those who would create databases for customization purposes, MySQL is the database programming language. Using the Matrix Database commands discussed in Chapter 6, linking and invoking any additional databases is very straightforward.

People experienced in Linux typically write shell scripts to perform a variety of functions. The Matrix Plan Editor provides the capability of incorporating shell script commands as well as Linux commands (e.g., copy, move, etc.) into the application via the LINUXCMD command under the *General* command category.

In the same light, there are users who are C and C<sup>++</sup> programmers. The Matrix software allows these programmers to write C and C<sup>++</sup> hooks under the same command category using the HOOK command. When hooks are added to an application, the executable must be compiled to include the hook and its return codes. Compiling is not required with the LINUXCMD command. Based on the capabilities discussed in this section, the power of Matrix<sup>2</sup> is limited only by the imagination of the user.

**MCCT IVR Failure Rate**

MCCT is presenting its MTBF results on the Matrix<sup>2</sup> IVR systems. We selected to do this with a Failure Rate analysis on a cross-sectional sample of our installed base. This approach is necessary since the manufacturers who are our source for major components (Aculab Voice Boards, Trenton CPU's, Adaptec, etc.) do not have MTBF figures on their products. In conversations with them, they have universally responded that the MTBF numbers are calculated predictions and not necessarily representative of real failure forecasts.

Given these facts and comments, we have looked at actual failures on an operational set of IVR's and are providing the historical hours for hard failures. The number of IVR systems in this set is 23. The table below represents the failure rate in this set.

<b>Major Component</b>	<b>Quantity</b>	<b>Hours of Operation</b>	<b>Failures</b>
Aculab Boards	73	2,517,400	1
Hard Disk Drives	42	2,035,600	3*
CPU's	23	2,081,400	1
Adaptec Controllers	17	2,212,200	1

On a "per system" basis, the number of operational hours for a failure to occur are:

<u>Component</u>	<u>Hours before Failure</u>
Aculab Boards	1,529,191
Hard Disk Drive	293,270
CPU	481,800
Adaptec Controller	306,600

\* Seagate had a high failure rate on their Barracuda line of Hard Drives for part of this period; some were detected before shipment and during run-in tests, some occurred after shipment.